SFA Modernization Partner

United States Department of Education

Student Financial Assistance

Integrated Technical Architecture
Detailed Design Document

Volume 6 – Development Architecture

Task Order #16

Deliverable # 16.1.2

October 13, 2000

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1. Purpose

The Development Architecture defines the development tools, methods, standards, and procedures that define the development environment for the Integrated Technical Architecture (ITA). The purpose of the development architecture is to support the tasks involved in the analysis, design, construction, and maintenance of the Student Financial Assistance (SFA) business applications.

1.2. Scope

This document will address the tools defined as part of the development environment for the ITA. The standards, guidelines, and procedures for using the tools as part of a standard SFA methodology are not included in this document.

1.3. Approach

The approach used to define the development architecture for the ITA was based on an analysis of the existing SFA development tools in place at the Virtual Data Center (VDC) and ongoing development tasks. In concert with these existing tools the development tools required to develop applications within the ITA environment were identified as well.

1.4. Document Organization

This document is organized into the following sections:

- Section 1 - contains an introduction of the development architecture and the scope of this document.
- Section 2 - provides an overview of the development architecture, the development tools and the mapping of each tool within the development environment framework.
- Section 3 – provides the development tools includes in the ITA Internet architecture.
- Section 4 – provides the development tools included in the ITA Data Warehouse architecture.
- Section 5 – provides the development tools included in the ITA Enterprise Architecture Integration(EAI).
- Section 6 – provides a description of the existing tools included in the development architecture currently installed or in operation at the VDC.

2 Development Architecture Overview

The Development architecture provides an environment for component-based solutions that support the Analysis, Design, and Construction phases of the development process. It is the combination of development tools, methods, standards, and procedures essential to a comprehensive, integrated environment for developing and maintaining systems. The development architecture provides a starting point for designing and building a development environment, and identifies key concepts and components for the environment.

2.1. Development Framework

The SFA Development Framework is based upon an Integrated Development Environment Architecture (IDEA). IDEA provides a development environment framework and associated guidelines that reduce the effort and costs involved with designing, implementing, and maintaining an integrated development environment.

The development environment is built upon an integrated set of tools and components, each supporting a specific task or set of tasks in the development process.

Figure 1 shows the central component, System Building, which is supported by eleven management components

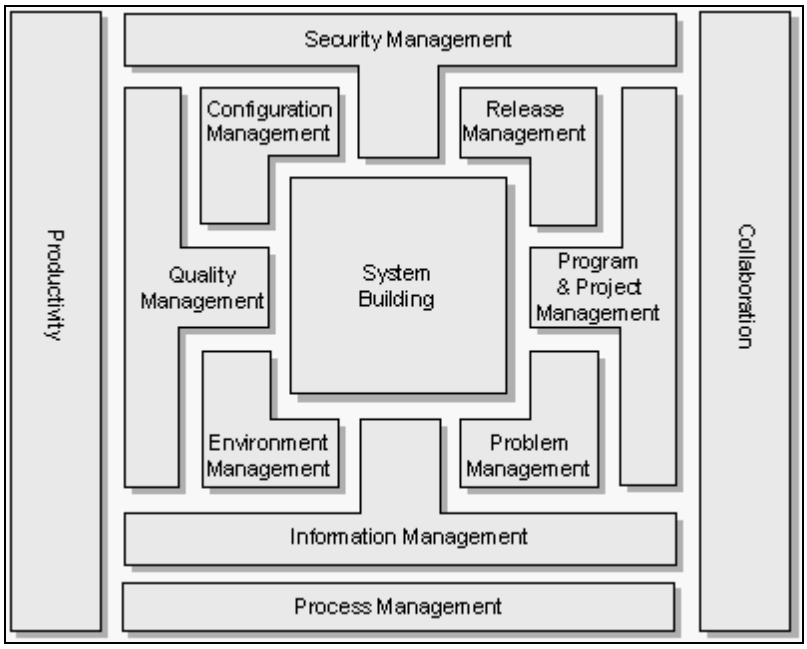

Figure 1 - Integrated Development Environment Architecture (IDEA)

A brief description of the services provided by Development Architecture is listed in the table which follows.

Table 1 – Development Architecture Services

Development Architecture Component	Description
Information Management Tools	Manage the information that supports the entire project – information that is used both in systems building and in other management processes.
Security Management Tools	Enable the development of security components.
Quality Management Tools	Ensure that an agreed-on level of quality in the system is reached. They are also used to provide information and process for improving the quality over time.
Program and Project Management Tools	Assist the management teams in their daily work.
Environment Management Tools	Comprised of the following tools to support Environment Management in the development environment. Change Management – Supports the various aspects identifying and managing change in the development environment, the key tool is the Data & Software Distribution which enables automated distribution of data and software to the workstations and servers of the development environment. Service Management – Supports various aspects of supporting and managing the interface with the developers. Service Planning – Planning required to anticipate and implement changes to the other areas: service management, systems management, change management and strategic planning. System Management – Supports the various aspects of supporting and managing the operation of the distributed system
Release Management Tools	Manages the simultaneous development of multiple releases.
Configuration Management Tools	Covers the version control, migration control and change control of system components such as code and its associated documentation.
Problem Management Tools	Pertain to the problem tracking and solution process.
Productivity Tools	Productivity tools provide the basic functionality required to create documents, spreadsheets, and simple graphics or diagrams. Personal Productivity tools are typically packaged as integrated suites of software. These packages provide the basic functionality required to create documents, spreadsheets and simple graphics or diagrams. More recently, the ability to access the Internet and browse electronic documentation has been added to the suite of Personal Productivity tools. - Spreadsheet - Graphics - Word Processor
Collaborative Tools	Enable groups of people to communicate and to share information, helping them work together effectively, regardless of location.
Process Integration Tools	Enforce the correct sequencing of tasks and tools in conformance with a pre-defined methodology.

2.2. System Building

Systems development tools are the core of the development environment. These are the tools used by the development team to capture system requirements, functional designs, detailed designs, as well as the detailed coding and testing information to manage the resulting development effort.

2.2.1. Analysis & Design

Analysis and design tools are used to specify the requirements for the system being developed. They are typically modeling and diagramming tools, which provide the ability to diagram system requirements and specify "what" a system must do. Design tools are used to specify "how" a system will implement these system requirements. They are typically diagramming tools, which graphically depict how the system will be built in terms of its key components.

- **Application Logic Design** – provides the capability to graphically depict the logic of the application
- **Communication Design** – allow for detailed design of each module and to lay the basis for more refined performance modeling
- **Component Modeling** – covers both designing components from scratch and customizing and integrating packaged software
- **Database Design** – provides the capability to graphically depict the database design for the system
- **Data Modeling** – provides the capability to graphically depict the logical data requirements for the system
- **Event Modeling** – provides the capability to graphically depict the events and associated responses for the system
- **Object Modeling** – provides the facility to model objects
- **Performance Modeling** – supports the analysis of performance over the network
- **Presentation Design** – provides the capability to design the presentation layer of the application
- **Process Modeling** – provides the capability to graphically depict the business functions and processes being supported by a system
- **Prototyping** – allows the interface and functionality to be tested during the design stage
- **Reuse Support** – manages assets for future reuse
- **Usability Testing** – ensures the final system is usable

2.2.2. Build

Construction (Build) tools are used to program or build the application adapters and connectors. The more custom development that must be performed, the more the complexity increases. The construction environment must support the development of business rules, data transformation logic and the capturing of events from the end source application.

- **Source Code Editor** – used to enter and edit source code for the application
- **Compiler/Linker/Interpreter** – converts source code to executable code
- **Source Code Debugger** – provides the capability to unit test a program
- **Generation** – automated tools which generate application components, for example: source code, windows, reports, shells, make files, include files, help text / module descriptions, trace code
- **Quality Assurance (QA) Utilities** – verify the quality of completed code
- **Code/Object Libraries** – provide the developer with ready-made components (such as graphical user interface (GUI) components or simple utilities), which may be integrated into the architecture or application code
- **Media Content Creation** – provide facilities to create and manipulate media content

2.2.3. Test

Test functions are performed on the developed application as a necessary step prior to the deployment to production. The test functions validate the developed software against the defined requirements developed during the analysis and design phase. The test phase verifies that the developed application functions as designed, meets requirements, and performs as designed in the production environment. In addition, the test functions perform regression testing to ensure that the introduction of new software or applications does not introduce any errors or impact operations of existing applications.

- **Emulation** – emulate components that are part of the target environment but are not in the development environment
- **Performance Management** – supports performance testing
- **Test Coverage Management** – used to document which parts of each program have been executed during the test
- **Test Data Management** – allows developers to create and maintain input data and expected results associated with a test plan
- **Test Data Manipulation** – enables direct editing of data values in files and databases
- **Test Execution** – supports and automates system tests
- **Test Planning** – includes test schedule, test execution tracking, test cycles, test scripts, test conditions, test condition generation, input data, and expected results

- **Test Result Comparison** – compares expected and actual results
- **SIR Management** – helps track each system investigation request from problem detection through document resolution

2.3. Development Architecture Tools Mapping

The following tables map the development environment tools currently in use or scheduled to be deployed in the SFA development environment. Each development architecture component identifies the set of tools within the defined framework.

Table 2 –SFA Development Environment Tools

Information Management Tools	Security Management Tools	Quality Management Tools	Program and Project Management Tools	Environment Management Tools	Release Management Tools
CCC / Harvest	BSAFE	None	Rational Suite	MicroStrategy Administrator	Rational ClearQuest
Endeavor	BMC Control SA		Rational ClearQuest	Rational ClearQuest	
Interwoven TeamSite	NetMonitor		Rational ClearCase	Rational ClearDDTS	
	Checkpoint Firewall-1		Microsoft Project		

Table 3 – SFA Mangement Tools

Configuration Management Tools	Problem Management Tools	Productivity Tools	Collaborative Tools	Process Integration Tools
Rational ClearCase	Rational ClearDDTS	Microsoft Word	Microsoft Outlook	Microsoft Project
CCC / Harvest		Microsoft PowerPoint		
Visual Source Safe (VSS) System		Microsoft Excel		

Table 4 – SFA Design Tools

Requirements Analysis	Design	Build	Test
Rational Suite	Rational RequisitePro	IBM Visual Age for Java	Rational TestStudio
AnalystStudio	Informatica Designer	Inprise Jbuilder	Rational PerformanceStudio
Rational RequisitePro	Interwoven TeamSite	WebSphere Studio	Informatica Server Manager
Rational ClearQuest	Microstrategy Architect	Informatica Repository Manager	CCC / Harvest
	SterlingCOOL:GEN	Informatica Designer	Autonomy - API Test
	Powerbuilder	Autonomy Tools	
		- Query / Indexer Module	
		- Import Module	
		- Web Spider Module	
		- User Agent Module	
		- Categorizer Module	
		- Agent Query Module	
		- HTTP API commands	
		Microstrategy	
		- Architect	
		- Agent	
		- Administrator	
		- Web page editor	
		- XML / XSL editor	

2.4. Development Architecture Configuration Diagram

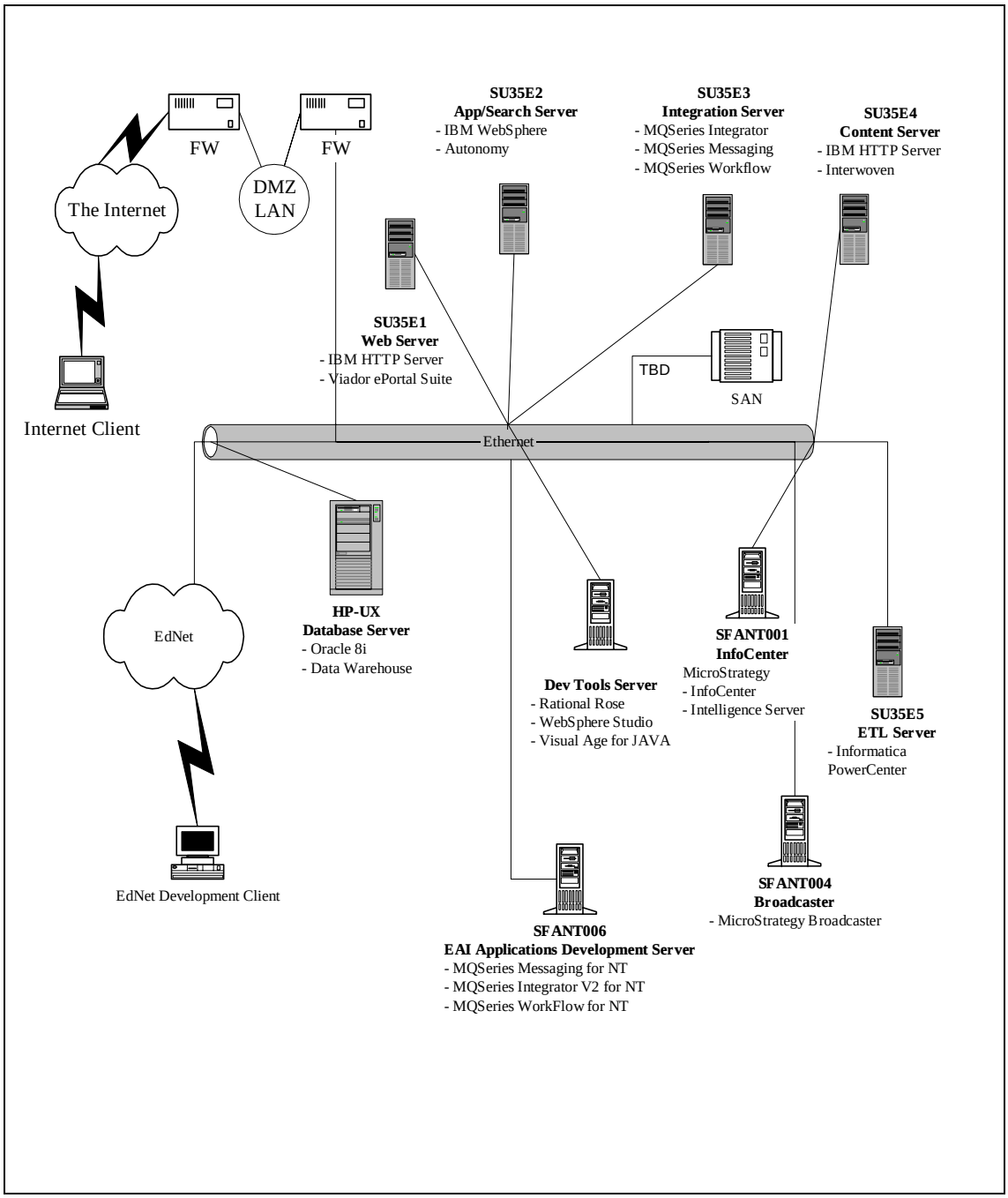

Figure 2 - ITA Development Architecture

The architecture to support the development environment of applications in the Integrated Technical Architecture is specified in the figure above. This environment consists of the development tools required for application development in the Integrated Technical Architecture. All development machines are located in the VDC and access is controlled

through EdNet or through secure access over the Internet or via dial-in. Once connected to EdNet each developer will have access to those VDC resources for which they have accounts and permissions. The exact configuration of each developer client workstation is based on their role and responsibilities. Sections 3-5 will define the client configuration requirements for developing applications utilizing the Integrated Technical Architecture product components. The following table defines the hardware and products deployed in the development environment.

Table 5 – Development Environment Hardware and Operating Systems

Development Hardware Configurations	Operating System
SU35E1-E5 - Sun E3500 - 4 CPUs - 4 GB RAM - 366 Mhz Processors	Solaris 2.6
SFANT001, 004, 006 - Compaq 1850R - Dual Processor - 1 GB RAM	NT 4.0 w / SP 5
HP N-Class Machines	HP-UX

2.5. Tool Selection Recommendation for SFA

In addition to the development tools included in the Integrated Technical Architecture development architecture, SFA has selected additional tools to provide full software life-cycle development support. These tools recommendations and selection were based on the following report: **Tool Selection Recommendation for Department of Education** *Vendor and Product Summary* December, 1999 Prepared by the Giga Information Group.

The tools referenced in this report and subsequently selected by SFA are referenced in this document as appropriate.

3 Internet Architecture Development Environment

This section defines the development tools as deployed to support the system building component of the framework as part of the Integrated Technical Architecture. The tools defined support the development of applications within the Internet Architecture framework.

3.1. Interwoven

TeamSite allows the user to structure Web development into different branches. Each branch contains private workareas, which contain complete virtual copies of the Website; a staging area, where contributors integrate their work; and editions, which are read-only snapshots of the Website at various points in its development cycle. Content is submitted from workareas to the staging area, and the staging area is then published as an edition. Editions may then be deployed to your production server or an alternative delivery channel as required.

The following two products are used by the developers / authors in order to contribute content to the Internet / Intranet:

TeamSite

Interwoven TeamSite Version 4.2.1 enables web developers to work in an environment that supports versioning of file system and database assets. The control of the assets contained within the TeamSite repository is accomplished by an easy-to-use workflow engine. The following features are provided:

- Utilization of branch structures to organize workgroups and enforce security
- Definition of Workareas or sandboxes specific to each content developer
- Versioning control of all deployed content

Templating

TeamSite Templating is an add-on package that enables decentralized content contribution, even in environments with centralized control of the overall look and feel. Templating provides the following benefits:

- Provide users the ability to submit content that can be stored directly in a database or a file, while preserving the integrity of Web page layout site architecture.
- Ability to create dynamic Web sites. Common content elements can be combined with data stored in a database or file system to deliver dynamic Web pages.
- Enables content reuse by providing the ability to include individual content elements in any number of Web pages.
- Administration of a consistent look and feel across the Web site, while maintaining the flexibility needed to make rapid changes.

All TeamSite content is maintained on a centralized Storage Area Network (SAN) at the VDC in defined content developer workspaces. Once content has been tested and validated in the development environment it is migrated into a defined Test environment. Upon acceptance testing in the Integration test environment content is then deployed to production for general availability. The flow of content is controlled through the TeamSite workflow engine.

3.1.1. TeamSite Development Architecture

The diagram below depicts the specific components and architecture required for TeamSite content development and administration.

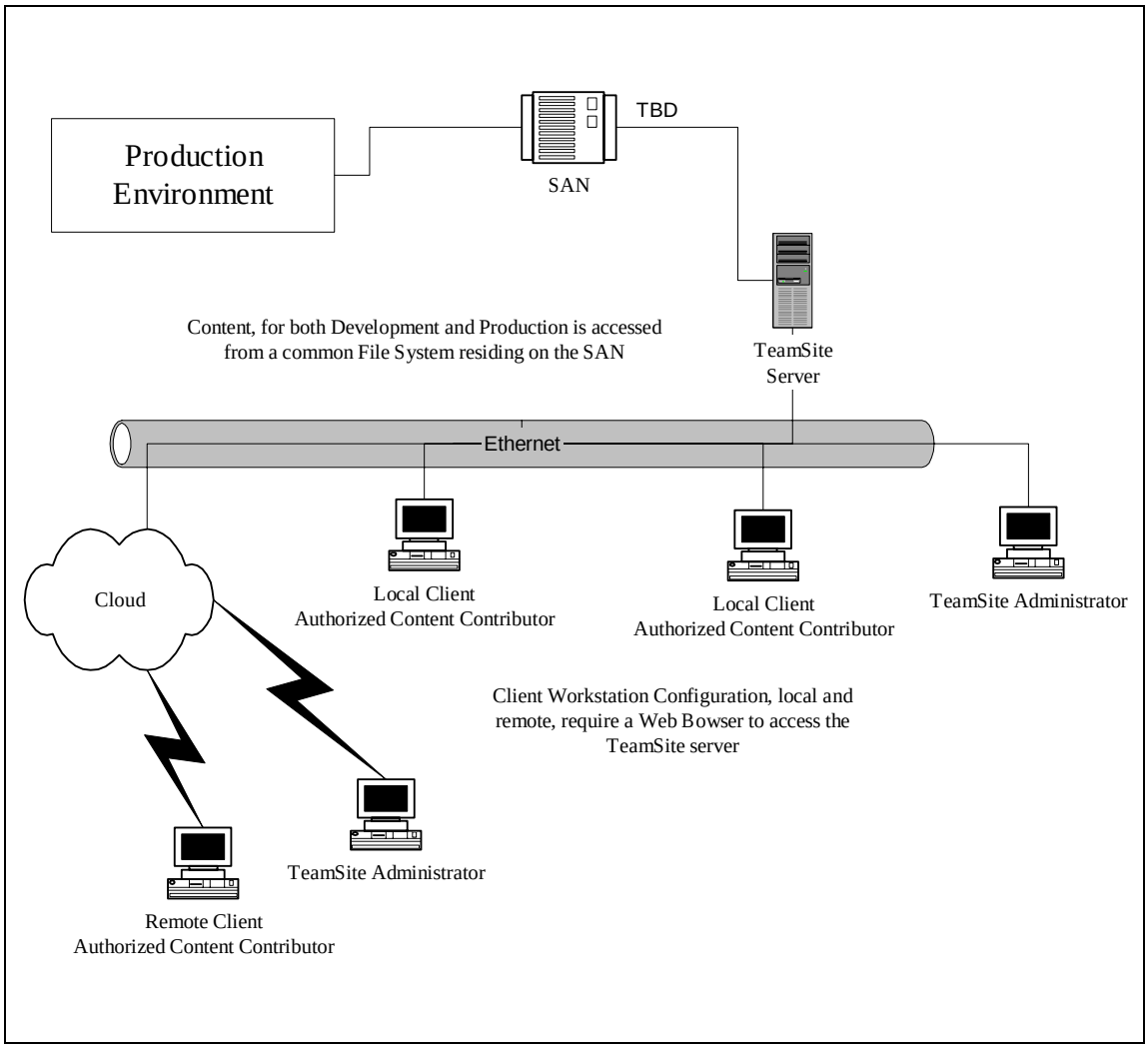

Figure 3 - TeamSite Development Architecture

3.1.2. TeamSite Architecture Requirements

TeamSite Server Requirements

- SUN E3500
- GB RAM
- TCP / IP Connection

The TeamSite Global Report Center is also installed and configured on the TeamSite server. It requires approximately 5 MB of physical memory. The OpenDeploy Global Report Center has the same requirement. Since both components are part of the SFA Integrated Technical Architecture architecture an additional 10 MB of physical memory must be allocated for both TeamSite and OpenDeploy.

Disk Space Requirements

All servers must have enough disk capacity for 760 MB of TeamSite program files (/iw-home) and five to ten times the total amount that the website content files is expected to consume (/iw-store). This amount of disk space is required in order to store TeamSite metadata and multiple versions under development website files.

Inode requirements

TeamSite requires a large number of inodes. To estimate how many inodes the server will require, use the following formula:

> # inodes = (# branches)(# average files in staging area per branch)(# average historical versions / file) (3 + 3(% of files having extended attributes) / 100))(safety-factor)
>
> For example, if the TeamSite server has three branches, with 20,000 files in the staging area of each branch, (on average), ten versions of each file in its history list (on average), seven percent of files have extended attributes, use a 1.5x safety factor:
>
> # inodes = 3 * 20,000 * 10 * (3 + 3*.07)* 1.5
>
> = 600,000 * 4.8
>
> = 2.9M inodes

Global Report Center Data Storage Requirements

The TeamSite Global Report Center requires an additional 25 MB of disk space, plus 10-50 MB for data storage.

TeamSite Software Requirements

The TeamSite server application runs on the same system as the website development server. It is recommend that the website development server be configured as a dedicated server,

running no applications other than the web server software and TeamSite. The following software is installed and meets or exceeds requirements:

- Sun Solaris 2.6 (recommended).
- International Business Machines (IBM) Hypertext Transfer Protocol (HTTP) Server or equivalent Apache version.

TeamSite Client Requirements

End users access TeamSite through browser-based thin-client technology. The only hardware requirements for client systems are the random access memory (RAM), central processing unit (CPU), local storage, and networking capability needed to operate a suitable web browser and the editing applications of the user's choice. TeamSite's thin-client interface does not require installation of any other client software unless you will be editing files through the TeamSite GUI.

The following table shows compatibility for most popular browsers on Solaris, 95, 98, and NT:

Table 6 - Browsers

	Netscape 4.6-4.6	Netscape 4.5x	Netscape 4.7	IE 4.x	IE 5.x
TeamSite GUI	Yes	Yes	Yes *	Yes	Yes *
Smart Context Editing	Yes	Yes	Yes	Yes	Yes
InterWoven Merge	Yes	Yes	Yes	Yes	Yes

* Recommended

3.1.3. Interwoven Product Architecture Overview

Private Workareas

With TeamSite, each Web contributor has a private workarea residing on the server to develop and modify Web content. Each workarea contains a virtual copy of the entire Web site. This allows each Web developer a place to stage and test changes in the context of the entire site without impacting the production site / content or other contributors' work areas. The figure below provides a diagram of the TeamSite content development flow and process.

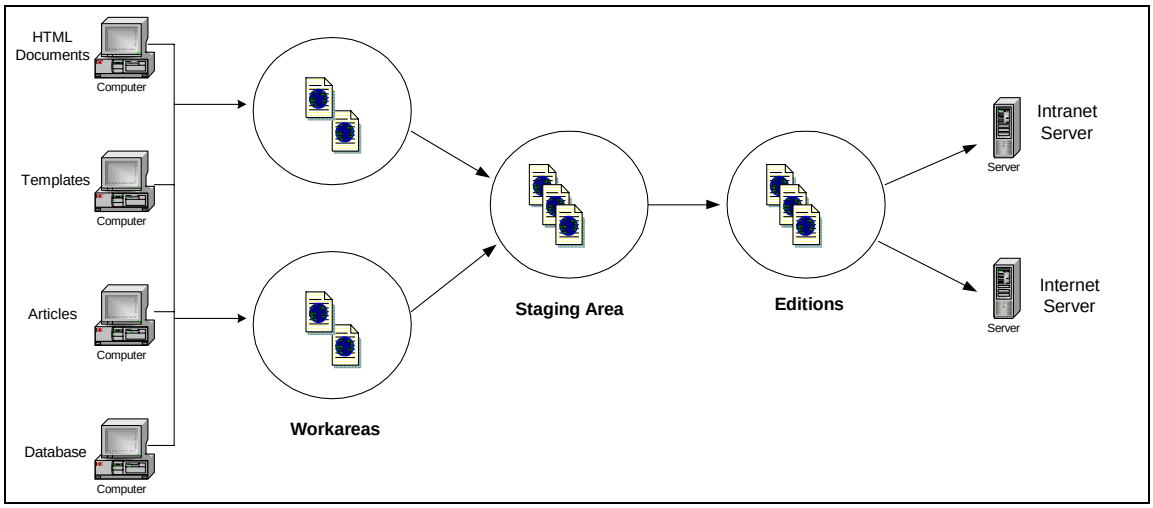

Figure 4 –TeamSite Workflow

Web contributors for the SFA Intranet and Internet will place their content into TeamSite. Workflows can follow many different paths, but the one depicted above is the most likely scenario to demonstrate content management and how it is deployed to the respective web sites.

Intelligent File System

TeamSite's Intelligent File System is composed of the TeamSite server and kernel, the TeamSite backing store of files and metadata, a suite of command-line tools, TeamSite CGI, proxy servers for access through the TeamSite browser-based GUI, and file system mounts for access through the file system interface. All of these components are included in the TeamSite development server installation or are services provided by the VDC.

The Intelligent File System is the core of the TeamSite system, where detailed information about the website, the web assets, web asset metadata, the production process and the users is stored. The Intelligent File System collects and maintains metadata on TeamSite files, directories, and areas, and allows TeamSite to process and present information according to who is asking for the information, and under what conditions. By using an object oriented design within a file system architecture, TeamSite combines extensive metadata tagging with open access and file system performance for web content.

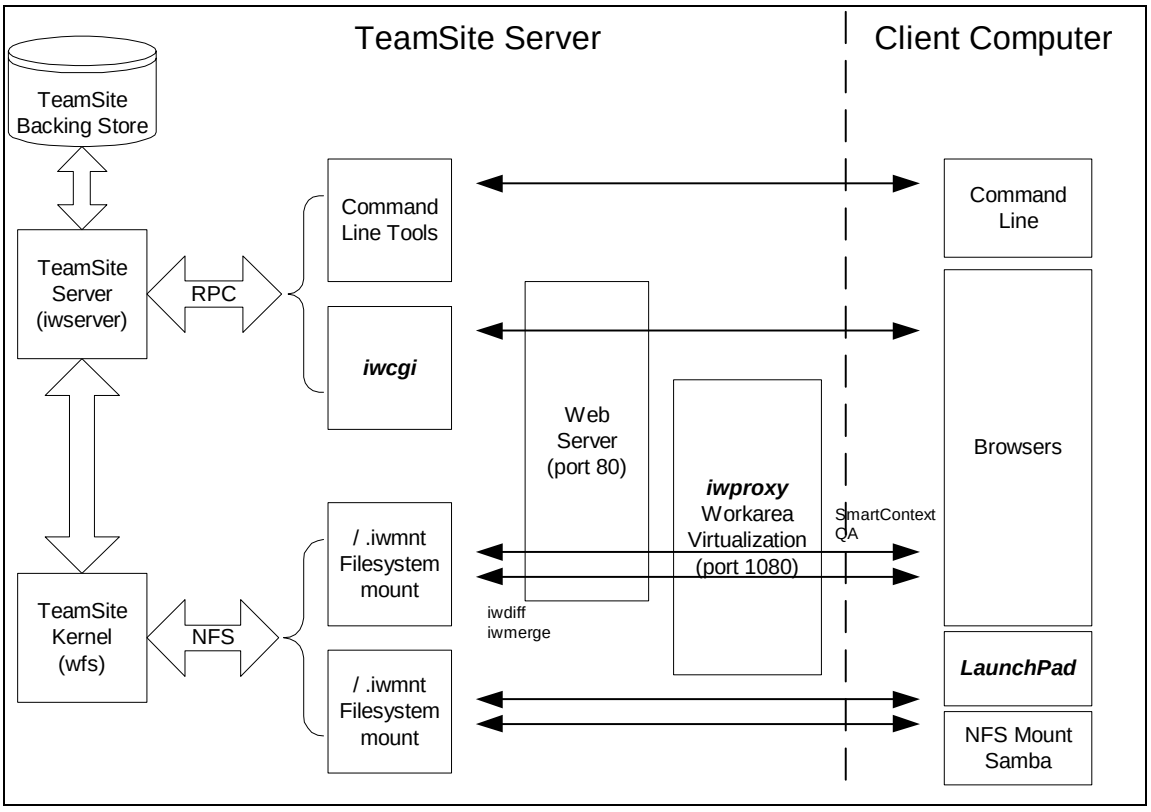

Figure 5 – TeaSite Server Interfaces

The client computer connects to the TeamSite server in several ways. Requests from the browsers or LaunchPad are routed through the standard TeamSite proxy server, which allows consistent views of TeamSite areas. The double proxy server redirects hard-coded links within the website. Requests through the file system interface (NFS mount / Samba) and command-line tools, which do not go through the webserver, are not routed through a proxy server.

The previous diagram depicts the internal access mechanisms and flow for accessing the TeamSite server via a client workstation for development. Some of the specifics may be modified to reflect the exact VDC configuration. For instance all access to file systems may be through AFS mounts rather than NFS.

3.1.4. TeamSite Templating Overview

Developing new templates requires a strict file and directory structure in order to get them to work. This is known as the data storage Hierarchy.

TeamSite Templating uses a data storage hierarchy based on data *categories* and *type*s. The directory structure supporting this hierarchy resides in the workarea for each TeamSite Templating user. The directory structure is as follows. Items in boxes are directories; items not in boxes are files.

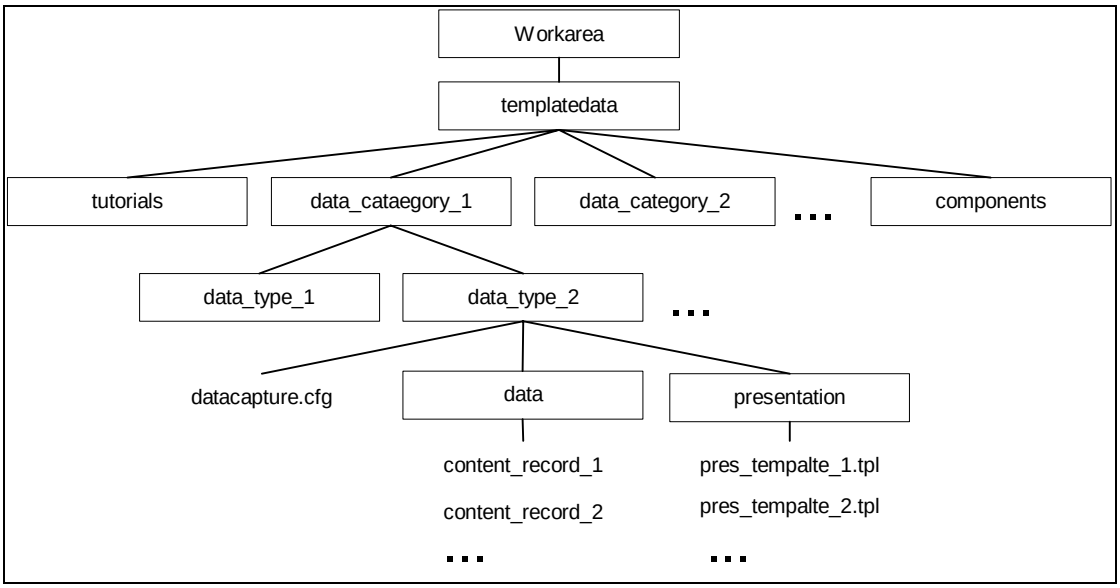

Figure 6 – TeamSite Templating Directory Structure

As shown, the templatedata directory is at the highest level in the hierarchy. Data categories are at the next level in the hierarchy and contain one or more data types. For example, the data category beverages could contain separate directories for the data types tea, coffee, milk, etc. In addition to residing in this directory structure, data categories and types must also be listed in the templating.cfg configuration file to be made available to TeamSite Templating. The component directory that stores component templates is also a subdirectory of templatedata.

Data type directories each contain a datacapture.cfg file and the subdirectories data and presentation. Details for the entire hierarchy are listed in the table that follows.

Table 7 – TeamSite Directory Hierarchy

File or Directory	Description
templatedata	Top-level directory containing subdirectories for data dategories, types, and all associated configuration files. Residesin the workarea for each user who uses TeamSite Templating. Can be renamed.
data_category_1	The first major categorization for data on a specific branch. Named and defined in templating.cfg. For example: /templatedata/beverages
data_type_2	The first subcategory of data in data_category_1. Named and defined in templating.cfg. For example: /templatedata/beverages/tea Each data type in a given data category has its own subdirectory.

File or Directory	Description
datacapture.cfg	The XML configuration file that defines a data capture template and drives data capture for a specific data type. As such, it defines the data type itself For example: What information the data type will contain, parameters for what type of data is legal in any input field, etc.) Specifies the look and fell of the data capture form displayed in the TeamSite GUI through which a content contributor enters data. Each data type must have exactly one datacapture.cfg file.
data	The directory containing all captured data content records for a given data type. If necessary, you can define and create a directory tree underneath the data directory. A data directory can contain zero or more data content records.
content_record_1	The first data content record for a given data type. Each data content record is an XML file containing formatting information interspersed with data that was captured from a content contributor via the TeamSite GUI. A data content record is named by the content contributor during data entry. For example: /templatedata/beverages/tea/data/november_order
presentation	The directory containing all presentation templates for a given data type. The presentation directory must contain one or more presentation templates.
pres_template_1.tpl	The first presentation template for a given data type. A data type can have any number of presentation templates. A single presentation template is populated by data from zero or one data content records. A presentation template can have a name of your choice. For example: /templatedata/beverages/tea/presentation/monthly_order.tpl
components	The directory where all component templates are stored.
tutorials	Examples showing the use of ix_xml tags.
data_type_2	A second subcategory of data in data_category_1. For example: /templatedata/beverages/coffee
data_category_2	A second major categorization for data on a specific branch. For example: /templatedata/food

3.1.5. TeamSite Elements

Branches

TeamSite provides *branches* for different paths of development for a website. Branches can be related to each other (e.g. alternate language versions of the same website) or they may be completely independent. Each branch contains all the content for a website.

A single branch contains archived copies of the website as *edition*s, a *staging area* for content integration, and individual *workareas* where users may develop content without disturbing each other. Branches can also contain *sub-branche*s, so development teams may keep alternate paths of development separate from each other. Content can be easily shared and synchronized across branches and sub-branches. Users may work on one branch or on several, and the number of branches on a system is not limited.

Branches facilitate distributed workflow because they allow separate teams to work independently on different projects. Because all branches are located on the same TeamSite server, it is easy for one team to incorporate the work of another into their project.

Workareas

Each *workarea* contains a virtual copy of the entire website, which may be modified in any way without affecting the work of other contributors. Users who have access to a workarea may modify files within that workarea and view their changes within the context of the entire website before integrating their work with that of other contributors. Users can lock files in each workarea, eliminating the possibility of conflicting edits.

All changes to files residing in a workarea are kept completely separate from other workareas and the staging area until the user chooses to promote his changes to the staging area. Within a workarea, users may add, edit, or delete files, or revert to older versions of files without affecting other users.

Staging Areas

Each branch contains one *staging area* where contributors incorporate their changes with the work of others. Users submit files from their workareas to the staging area to integrate their work with other contributions, and test the integrity of the resulting website. Because the staging area is an integrated component of the system, conflicts are easily identified and different versions of the same file can be merged, rather than overwritten.

Editions

Editions are read-only snapshots of the entire website, taken at sequential points in its development. Contributors can create new editions any time they feel their work is well integrated, or any time they want to create an update to the website for reference or deployment. Each edition is a fully functional version of the website, so users may see the development of the website over time and compare it with current work.

3.1.6. TeamSite Users

Authors

Authors are primary content creators. All work done by Authors goes through an explicit approval step. They can receive assignments from Editors, which are displayed in To-Do lists when Authors log in to TeamSite. Authors can access TeamSite from a simple browser-based interface, and do not need to be sophisticated computer users.

In order to test and QA work, Authors have full access to the content in their Editors' workareas, but do not need to concern themselves with the larger structure and functionality of TeamSite. The Author role is appropriate for non-technical users, or for more technical contributors who do not need access to TeamSite's extended functionality, such as TeamSite's advanced version management features.

Editors

Editors own workareas. They create and edit content, just as Authors do, but they are primarily responsible for managing the development within their workareas. This includes assigning files to Authors and submitting completed content to the staging area, and it may include publishing editions.

Editors have access to specialized TeamSite content and workflow management functions. Editors are generally "managerial" users, who primarily supervise the work of Authors, or self-managing "power" users, who need TeamSite's extended functionality to manage their own content.

Administrators

Administrators own branches. They have all the abilities of Editors, but they are primarily responsible for the content and functioning of their branch. Administrators can manage project workflow by creating new workareas for Editors and groups, and by creating sub-branches of their own branch to explore separate paths of development. An Administrator is the supervisor of the project being developed on his branch. The Administrator may be the webmaster for a particular version of the website, or a project manager.

Masters

Master users own the website. They can perform all the functions of Editors and Administrators on any branch. The Master user owns the main branch, from which all sub-branches are created. The Master user is generally involved in the installation of TeamSite, and can reconfigure TeamSite on a system-wide basis.

3.1.7. TeamSite Templating Model

TeamSite Templating's architecture allows data capture and data presentation to be configured, executed, and managed separately. The following diagram and sections provide a high-level overview of this architecture.

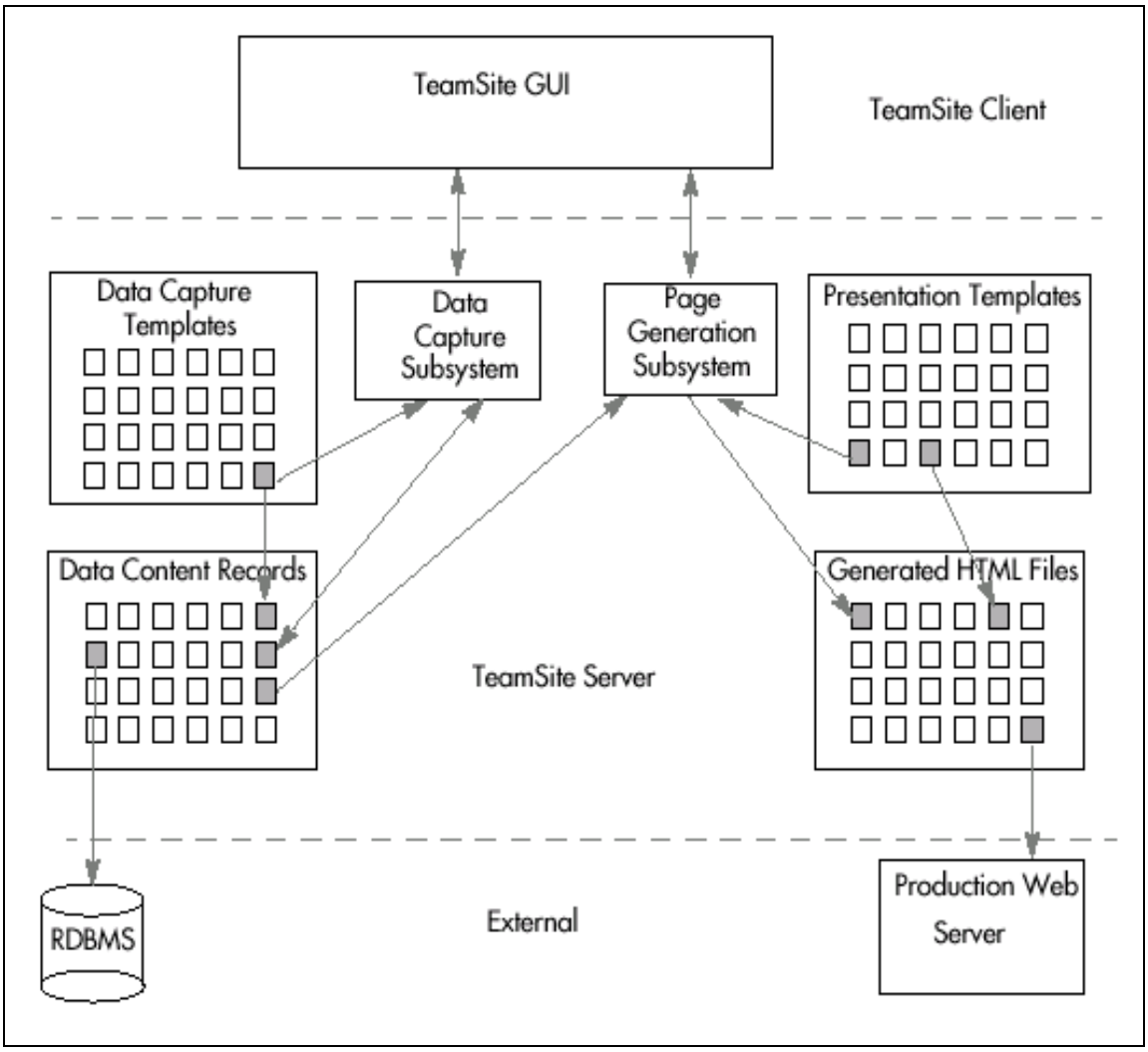

Figure 7 – TeamSite Data Deploy Architecture

Data Capture

Content contributors working through the TeamSite GUI have access to the *data capture subsystem*. This subsystem lets content contributors select and work through forms defined by *data capture templates* to create or edit *data content record*s, which by default are stored in the TeamSite file system. After data content records are created, they can be displayed via presentation templates or optionally deployed to a database via DataDeploy.

Data Presentation

After data is captured and stored as data content records, users working through the TeamSite GUI or from the command line can access the page generation subsystem to combine a data content record with a presentation template. The end result is a generated Hypertext Markup Language (HTML) file that displays the data content in a way defined by the presentation template. Additionally, users can generate an HTML file that obtains data

from zero or one data content records and from queries to databases. The generated HTML file can optionally be deployed to a production web server via OpenDeploy.

3.2. Viador

The Viador Portal Framework is "a portal-building" toolkit consisting of various shared portal services including the ability to access a potentially unlimited universe of data sources and applications. The framework enables developers to focus on quickly and easily building and integrating portlets into the portal framework, relying on the portal to deliver security, scalability, data access and other services.

The Viador E-Portal Framework gives the portal developer an infrastructure for creating, deploying, maintaining and managing an enterprise information portal environment. It includes the following components:

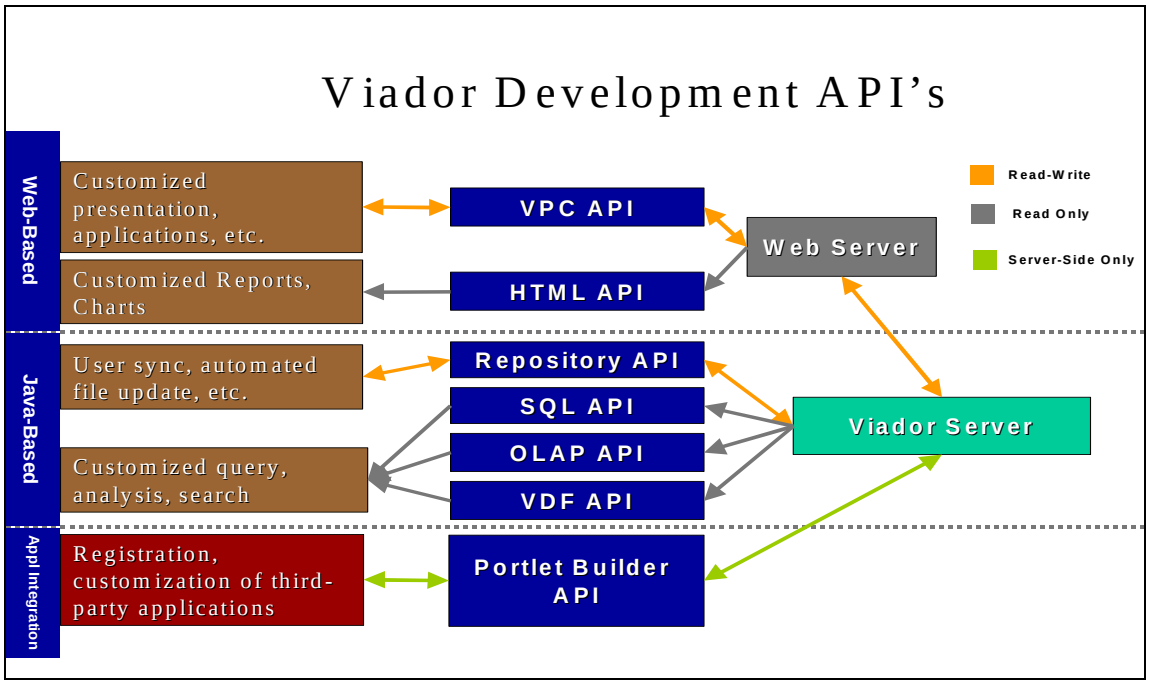

Figure 8 – Viador Development API's

3.2.1. Viador Portal Customization

Viador Portal Customization API

To customize the Viador E-Portal Suite, call a Factory Object to create and / or manipulate a Viador Object. Creating new Viador Objects by means of Factory Objects is the basis of the Viador Portal Customization (VPC). Factory objects are constructors used to create new instances of Viador objects. Factory objects use an applet to communicate with the Viador Server. The VPC Application Programming Interface (API) consists of JavaScript constants, objects and methods. These are defined in two JavaScript files,

...\infospc\scripts\object\ViadorFactory.js

...\infospc\scripts\object\ViadorObjects.js

HTML API

The Viador HTML API allows users to run report queries against relational databases. The HTML API can perform the following tasks:

- Run previously created reports against fresh data.
- Run parameterized reports with values for the parameters that you specify.
- Drill-across a report column value to find other data relevant for the particular column value.
- Create a report of all reports that a specific user can access.

Portlet Development

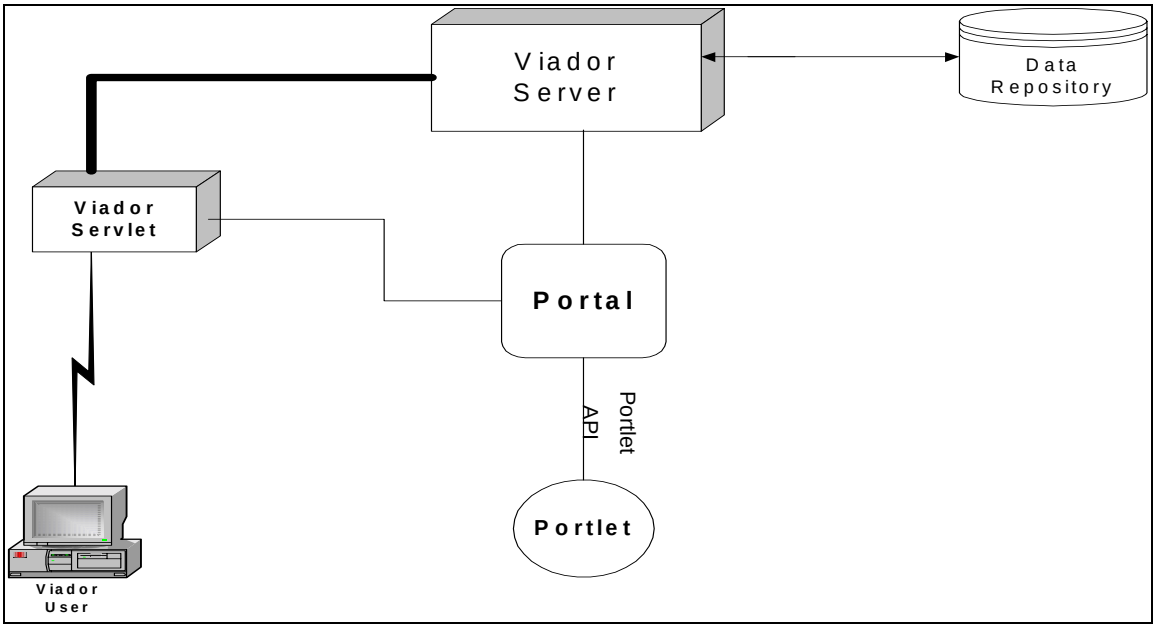

Figure 9 – Viador Pertlet Development

Portlet Builder API

The Viador E-Portal Framework includes the Viador Portlet Development Kit (PDK). The PDK provides the developer with access to a set of open, published portal APIs that let the developer:

- **Manage portal objects** - Create, modify, retrieve, or delete objects, and assign security profiles to objects.

- **Design composite portal pages without programming -** Build highly interactive, user-friendly, personalized portal pages based on a variety of technologies, such as Java and JavaScript APIs used to create and display dynamic HTML pages and forms.
- **Integrate third-party applications -** A set of APIs and applications that make it easy for portal developers to quickly and easily add new applications and portal content by registering the applications in the repository. This provides a clean, open mechanism for extending the portal to include packaged applications, desktop applications, legacy applications, and external applications (e.g., e-commerce).

3.2.2. Viador Server Platforms

Viador supports development on any of the following server platforms:

Table 8 – Server Platform Requirements

Product Version	Server Platform	Web Server
6.1.1	Sun Solaris 2.6	Netscape Enterprise Server 3.6 Apache 1.3.9

Client Platforms

Viador supports development on any of the following client platforms:

Table 9 – Client Platform Requirements

Product Version	Server Platform	Web Server
6.1.1	Windows NT 4.0, 95/98	Netscape 4.51, 4.61, 4.72 Internet Explorer 4.01 (4.72.3110.8) ** Internet Explorer 5.0

** Recommended

3.3. Autonomy

Autonomy's technology offers a breakthrough in managing unstructured digital information, including word processing and HTML-based files, email messages, and electronic news feeds. By applying sophisticated concept matching techniques to the problems of information access and distribution, Autonomy has created a set of products to automate the process of getting the right information to the right person at the right time. These products not only improve the efficiency of information retrieval, but also enable the dynamic personalization of digital content.

Autonomy's architecture combines innovative high-performance pattern-matching algorithms with sophisticated contextual analysis and concept extraction to automate the

categorization and cross-referencing of information, improve the efficiency of information retrieval and enable the dynamic personalization of digital content.

Autonomy's strength lies in its high-performance pattern matching algorithms. These algorithms are informed by Claude Shannon's principles of information theory, Bayesian probabilities, and the latest research in neural networks. This technique enables Autonomy's system to identify patterns in text and look for similar patterns in other sources, quickly and automatically. Most importantly, the technology can analyze a text and identify the key concepts within the document because it understands how the frequency and relationships of terms correlate with meaning.

Autonomy employs advanced pattern matching technology (non-linear adaptive digital signal processing) to extract a document's digital essence to determine the characteristics that give the text meaning. Once Autonomy's technology has identified and encoded the unique "signature" of the key concepts, Concept Agents are created to seek out similar ideas in websites, news feeds, email archives and other documents. Because it does not rely on key words, it can work with any language.

3.3.1. Autonomy Development Architecture

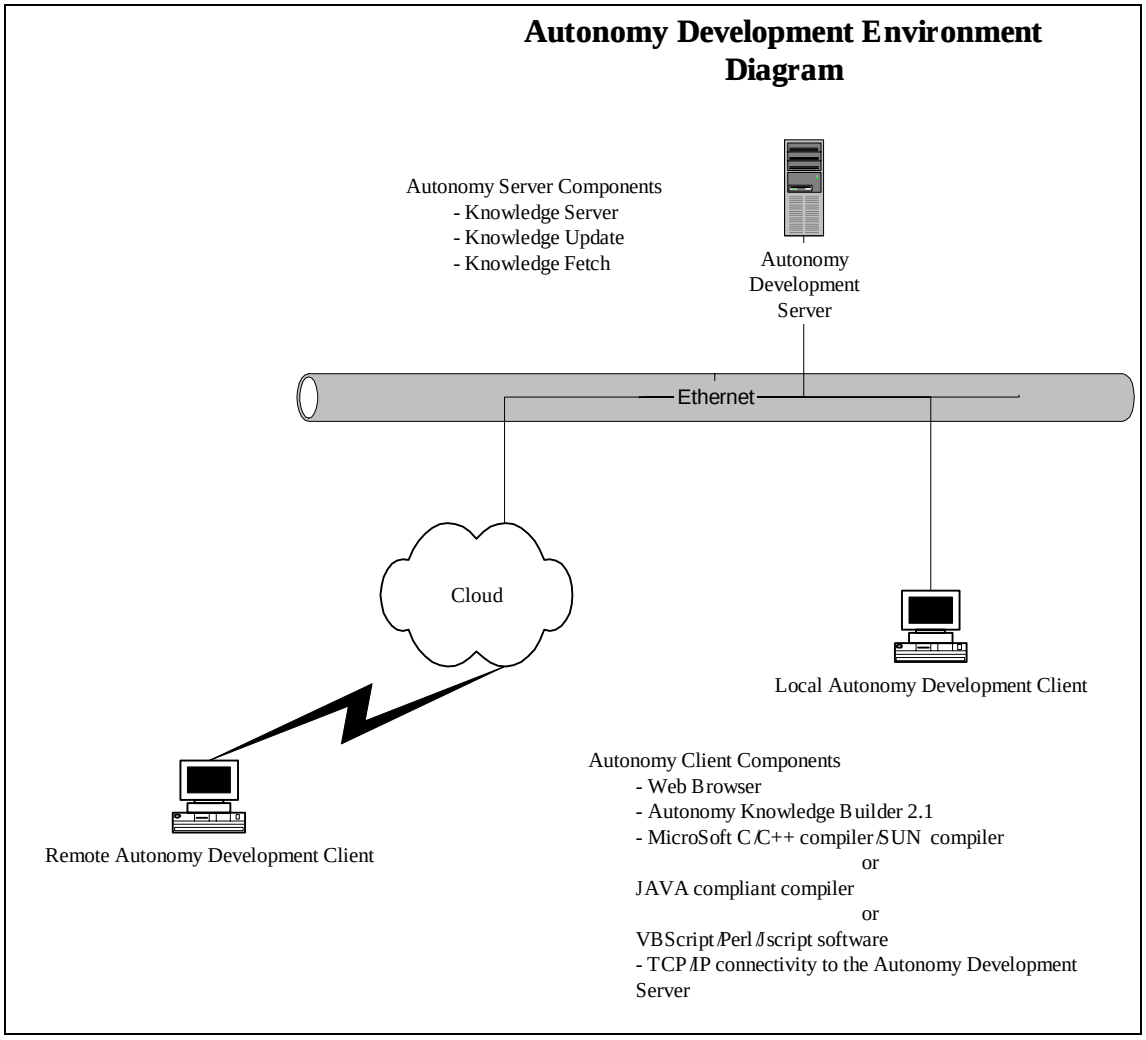

Figure 10 – Automoy Development

The development environment for Autonomy development is depicted in the figure below. This recommended hardware configurations for the server and client workstation is specified in the table following the table.

Table 10 – Autonomy Development Requirements

Hardware	Machine Type	Machine Function
Autonomy Server Solaris 3500 Dual Processor 550 MHz (Solaris) Dual Processor Pentium III 500 MHz (NT) 1GB RAM 30 GB disk space	SUN Solaris or Windows NT	Query Service Index Service Categorization Service
Autonomy Client Development Workstation Pentium 300 MHz 128 MB RAM 2 GB disk space	Windows 95 / 2000 / NT	To develop code using the Autonomy toolkit. The Autonomy API will access the Autonomy services on the Autonomy server through a TCP / IP communication.

3.3.2. Autonomy Tools Overview

The Knowledge Builder provides access to all of Autonomy's core functionality. This API has not been defined as an after thought, but has formed part of the product since the first release and is used by Autonomy's developers.

Autonomy's Knowledge Builder API:

- Enables user interface developers to provide an application look and feel that is productive and consistent with enterprise standards.
- Enables application developers to implement robust and efficient business logic effectively using familiar development tools.
- Enables OEM partners to integrate the full range of Autonomy functionality with their products.
- Enables data specialists to have fine-grained control over key corporate assets.

The diagram below illustrates how Autonomy provides the necessary features to enable enterprise integration.

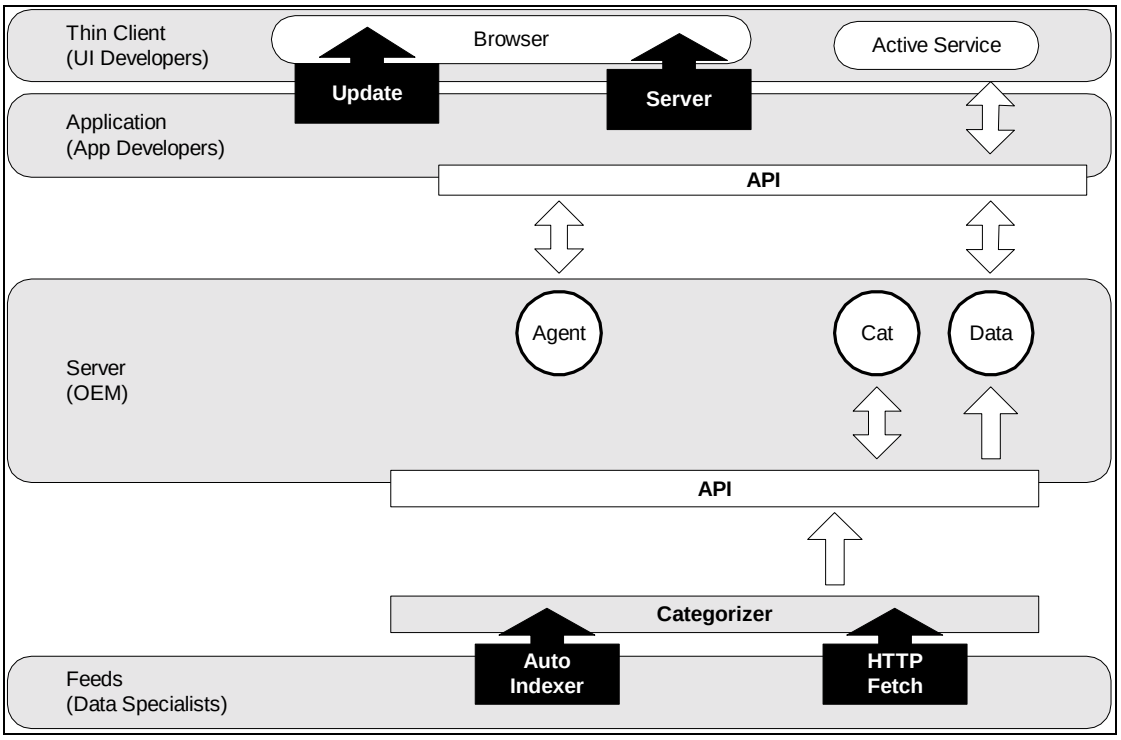

Figure 11 – Autonomy Enterpirse Integration

For the user interface developer, Autonomy provides HTML Templates and Active Server Pages (ASPs) to enable the development of application front ends using a range of tools including HTML editors or Java. Application developers are provided with Common Gateway Interface (CGI) scripts to enable the development of CGI compliant tools including Perl and Tool Command Language (TCL). Application developers can also use the API to develop applications in a wide range of languages, as shown in the following diagram. The use of HTTP to underpin the Autonomy API enables the development of robust, distributed applications.

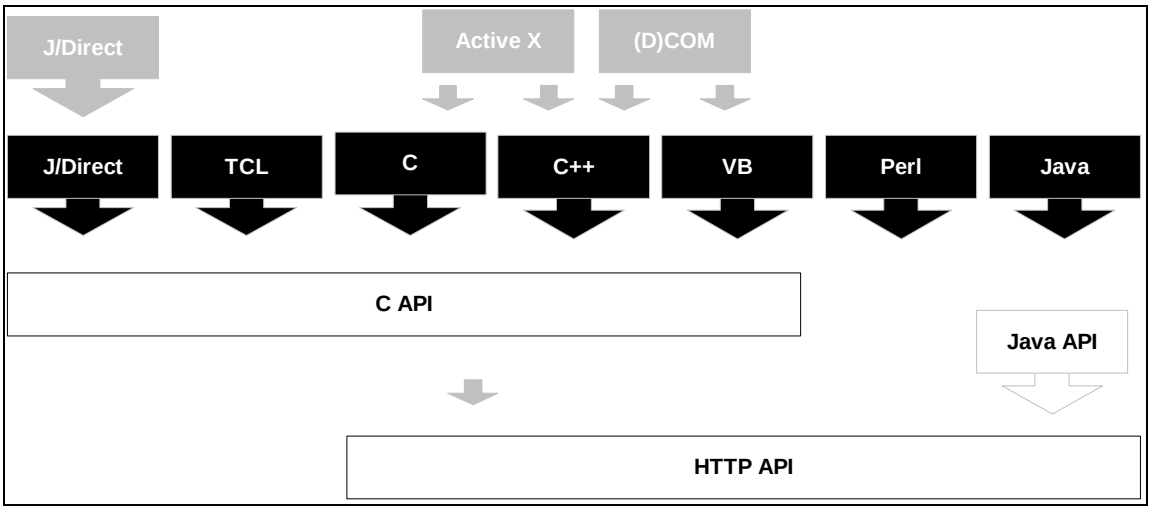

Figure 12 – Autonomy API Supported Programming Languages

3.3.3. Supported Programming Languages

The Autonomy API can be split up into three parts: The C API, HTTP API and Java API. All Autonomy Modules use the C API to communicate with each other however only commands used to communicate with the Dynamic Reasoning Engine (DRE) have been translated into HTTP requests and Java Commands. Both the C API and the Java API use the HTTP API to communicate with the DRE.

The Autonomy API can be called in three different ways, although the available functionality is almost identical in each case. In most cases the API is called using C function calls, however for some commands it is possible to use HTTP Requests and / or Java Classes to call the API.

The Autonomy import API and configuration files provide tools to build data processing applications to Autonomy-enable more than 200 data formats.

Table 11 – Autonomy development Tools

Type	Tool	Version	Framework Component
Query / Indexer development	Query / Indexer Module	2.1	Build
Import development	Import Module	2.1	Build
Web Spider development	Web Spider Module	2.1	Build
User Agent development	User Agent Module	2.1	Build
Categorizer development	Categorizer Module	2.1	Build
Agent Query development	Agent Query Module	2.1	Build
API Test	Http API commands	2.1	Test

3.3.4. Autonomy Query/Indexer Development

This module provides the developers with all the tools required to perform core querying and index functions.

Features

- Natural language, Boolean, Proximity, Agent and concept querying.
- Fast query speeds: the speed linked to the number of results requested rather than the number of documents indexed.
- Language independent.
- Query on date information.
- Support for configurable structured field information with additional Boolean field query options.
- Simple socket based interface to query engine, allowing use in custom applications from a wide range of applications.
- Multiple Concept indexes per engine.
- Automatic summarization of documents based on key concepts.
- Automatic suggestion of related documents / references within the databases.
- Retrieval of original document or its plain text content.
- Automatic extraction of key concepts.
- High indexing speed.
- Automatic elimination of duplicates by content or reference on indexing.
- Remote administration of indexing processes, management of Concept indexes.
- Optional indexing of additional field information.
- Optional storage of original content.
- Operates directly or through the distributed query handler.
- Open index file format allowing for simple development of new information feeds.

Functions

- Text / Fuzzy / Proximity query return results as flat file, buffer or structures.
- Suggest on a document or number of documents to flat file, buffer or structures.
- Retrieve key terms for a document.
- Retrieve document contents.
- Get intelligent / quick document summary of a specified length.
- Index document (remove duplicates by reference, content match or field match).

- Delete document (by ID or reference).
- Create database.
- Delete database.
- Reset Engine.

Technical Specifications

Allows custom applications to access ALL of the query and indexing functionality of the Autonomy DRE engine

Query:

- Concept / Boolean / Proximity / Propername / Wildcard
- Field restrictions
- Date restrictions
- Thesauri query

Indexing:

- Local files
- Remote files
- Real-time live indexing
- Engine administration
- Duplicate removal

3.3.5. Autonomy Import Development

Using this module, a developer can provide import support for numerous document types.

Features

- Support for over 200 formats including: HTML, plain text, word processing, spreadsheets, presentations.
- Support for many sources including: Intranet / Internet sites, File Servers.
- Intelligent selection of filter formats.
- Configurable extraction of field data from documents.

Functions

- Set-up the import parameters.
- Set-up the field extraction parameters.

- Import a set of files / directories using intelligent filter selection.

Technical Specifications

Provides access to basic KeyView and Portable Document Format (PDF) import functionality

- High performance import for HTML / text:
  - Link stripping and Meta Tag extraction
- Import Microsoft Word, Excel, PowerPoint, Rich Text Format, etc:
  - All document properties supported
- Import Adobe PDF format:
  - Breaking on page / paragraph boundaries
- Robust slave technology
- Fields:
  - Fixed tagging
  - String match extraction
- Size limits and word counts
- Flexible page layout / breaking parameters
- Quick summary extraction

Import Formats

The Import modules supports a wide range of document formats.

- Adobe
- Acrobat PDF format
- KeyView
  - Applix Words V4.2
  - Lotus Ami Pro V2.0, 3.0, WordPro 96 / 97
  - Microsoft RT, Word for Windows V2.x
  - Microsoft Word V6.0 / 95, 9,
  - Mac V4.x to 6.x, V2.0 to 5.5
  - Microsoft Works V3.0, 4.0
  - Microsoft Write
  - Unicode V2.0
  - WordPerfect V5.x, for Windows V6.x to 8.x

- Mac V2.0, 3.x
- XyWrite for Windows V4.12
- Various spreadsheets
- Various presentations, PowerPoint

- Basic
  - Plain Text
  - HTML

3.3.6. Autonomy Web Spider Development

The Spider module allows the developer to create applications that retrieve content from the Internet or Intranets. It provides conditional retrieval features and offers high performance.

Features

- Multiple socket connections for optimizing bandwidth usage.
- Simultaneous retrieval of multiple remote / local website pages.
- Conditional retrieval based on date ranges and configurable date formats.
- Conditional retrieval based on configurable inclusive and exclusive wild card lists, optionally checked in the URL, header and body.
- Optional 'politeness' delay between retrievals / retrieval rate.
- HTTP authentication, HTML forms, login, and HTML cookie authentication supported.
- Insertion of meta tags for import / index processes.
- Insertion of original reference to allow viewing of retrieved pages.
- Support for Proxy Servers and Firewalls.

Functions

- Set up the spider parameters.
- Set up the spider page criteria.
- Set up the spider Login parameters.
- Set up the spider proxy parameters.
- Launch a single spider.
- Launch multiple socket spider(s).

Technical Specifications

Allows collection of documents from Intranet and Internet sites

- Fast, robust HTTP spider
- Support for:
  - CGI Login
  - HTTP Authentication
  - Cookies
  - SSL sites
  - Proxy / Firewall and authentication
  - Robots Protocol Standard
- Intelligent Updating
  - Prediction and calculation of page changes
  - Auto-deletion of missing URLs
- Page Criteria, including:
  - Wildcard string matches
  - Size limits
  - Link limits
- Flexible date format definition
- Socket multiplexing efficient multi-site retrieval

3.3.7. Autonomy User Agent Development

This module provides developers with functions to store and retrieve user information and agents.

Features

- Creation / Modification / Deletion of users and user agents.
- Persistent storage of user information, agent information training and retraining in encrypted server files.

Functions

- Add user group to disk.
- Remove user group from disk.
- Get maximum permitted users in a group.
- Add user to group.
- Remove a user from a group.

- Read a user from disk.
- Write a user to disk.
- Read user's agent list from disk.
- Write user's agent list to disk.
- Create agent.
- Add agent to user.
- Retrieve an agent by name.
- Delete an agent by name.

Technical Specifications

Allows custom applications to use Autonomy's User and Agent repository

- Manage Groups, Users, Agents
- Hierarchical structure
- Encrypted file storage
- Cross Platform
- Scaleable
- Remote access
- Callable from Java or C

3.3.8. Autonomy Categorizer Development

The categorization API module provides categorization capabilities using a simple command set. The developer can use functions to create categories and then discover which categories a new document most closely matches.

Features

- Automatic categorization of new documents.
- Provision for tagging of categorized documents.
- Simple Category definition mechanism.

Functions

- Make Category.
- Categorize.

Technical Specifications

Provides the functionality needed to create, train and edit category agents and to perform categorization of documents with them

- Create / Add / Remove agents from live categorizer
- Specify training text / files or DRE documents
- Set agent categorize action parameters
  - Tagging (Including HTML, SGML and XML)
  - Filing (copy / move)
  - Document list generation
  - Emailing
- High performance categorization function

3.3.9. Autonomy Agent Query Development

The AgentQuery module exposes the agent query and index functionality - allowing developers to perform agent queries, produce communities of agents, live alert systems and retrain agents.

Features

- Query using agents.
- Retrain agents from query results.
- Comparison of agents.
- Testing of text against agents.
- Indexing of agents.

Functions

- Perform a query using an agent.
- Retrain an agent on document(s).
- Compare agents.
- Test text against an agent.
- Index agent.

Technical Specifications

Allows custom applications to utilize the Autonomy Agent query and community functionality.

- **Query** - Find similar agents Find documents with an agent Find agents with a document (alert)
- **Indexing** - Add / Remove agent from community

3.3.10. Autonomy API Test

The HTTP API commands have several advantages; firstly it can easily be used with almost any Internet enabled development package from C++ to Visual Basic. Secondly, functions can be called manually using a web browser, allowing the developer to rapidly check the responses that the DRE is returning and perform indexing commands manually.

3.4. IBM WebSphere and Visual Age for Java

3.4.1. WebSphere Introduction

WebSphere Studio

The WebSphere Studio is a suite of tools that brings all aspects of Web site development into a common interface. Content authors, graphic artists, programmers, and Webmasters can all work on the same projects, each having access to the files they need. WebSphere Studio provides the capability to cooperatively create, assemble, publish, and maintain dynamic interactive Web applications.

The Studio is composed of the Workbench, the Page Designer, the Remote Debugger, and wizards, and it comes with companion Web development products. WebSphere Studio enables developers to create interactive Web sites that support advanced business functions, including the following: Create Java beans, database queries, and Java servlets, using the Studio wizards. The wizards make it easy to produce the input forms, the output pages, and the Java code that makes it all work together. Web site files can be grouped into projects and folders based on what makes sense to the development organization. Included are filters and global search capabilities to find the files required for development. Utilizing objects with defined behavior developers can add speed and efficiency to routine tasks. Files can be maintained individually or in a shared version control system—this repository can exist locally, on each developer workstation, on other systems in your network, or in a full-function version control system (VCS). Developers can customize the editing and updating of files with preferred tools.

Studio WorkBench

The Studio Workbench provides the capabilities to manage and maintain Web site applications and files and provides the following capabilities:

- Graphical display of the link relationships between the files in a project
- Automatic updating of links whenever files change or move
- Ability to register multiple authoring tools

- Ability to stage your Web site production cycle and publish various stages to different (and to multiple) servers
- Provides an import wizard that simplifies the transfer of existing site content directly into a Studio project
- Archive Web sites or sub-sites in a single file
- Ability to easily integrate third-party tools right into the Workbench environment
- Provides an enhanced team environment with a common view of work-in-progress, through the integration of popular source control-management software such as IBM VisualAge ®, Microsoft SourceSafe ® , PVCS, Rational ClearCase

Studio Page Designer

The Studio Page Designer provides a visual design environment that enables developers to create JavaServer Pages (JSP), Java servlets, and other Java-based Web tools. The Studio Page Designer can also be used to create Dynamic HTML (DHTML) and HTML pages and includes the capability to easily edit and toggle between the HTML or DHTML source and the browser view.

Studio Remote Debugger

The Studio Remote Debugger provides source level debugging of JSP files and Java servlets within the Studio environment. Remote debugging is possible on any machine that contains WebSphere Application Server 3.0 or above.

3.4.2. VisualAge for Java Introduction

VisualAge for Java

An integrated development environment that supports the complete cycle of Java program development. Although it is not a component of WebSphere Application Server or the WebSphere Family, VisualAge for Java is tightly integrated with the WebSphere Application Server. This integration enables VisualAge developers to develop, deploy, and test their Java programs without leaving VisualAge. It also provides developers with an environment that helps to manage the complexity common in the enterprise environment and is capable of automating routine steps.

Developers can use VisualAge for Java visual programming features to quickly develop Java applets and applications. The Visual Composition Editor provides point and click capabilities to do the following:

- Design the user interface for programs
- Specify the behavior of the user interface elements
- Define the relationship between the user interface and the rest of your program

VisualAge for Java generates the Java code to implement what is visually specified in the Visual Composition Editor. In many cases programs can be designed and executed without writing any Java code. In addition to its visual programming features, VisualAge for Java provides SmartGuides to lead developers quickly through many tasks, including the creation of applets, servlets, applications, Java beans, and enterprise beans built to the Enterprise Java Beans (EJB) Specification. It also enables the importing of existing code and exporting code as required from the underlying file system.

VisualAge Generator

Provides the capability to deliver high-volume transaction processing in multi-tier, multi-platform client / server environments. VisualAge Generator masks the complexity of data and communications connections and is optimized for DB2, Customer Information Control System (CICS), IMS and WebSphere servers.

3.4.3. WebSphere/VAJ Development Architecture

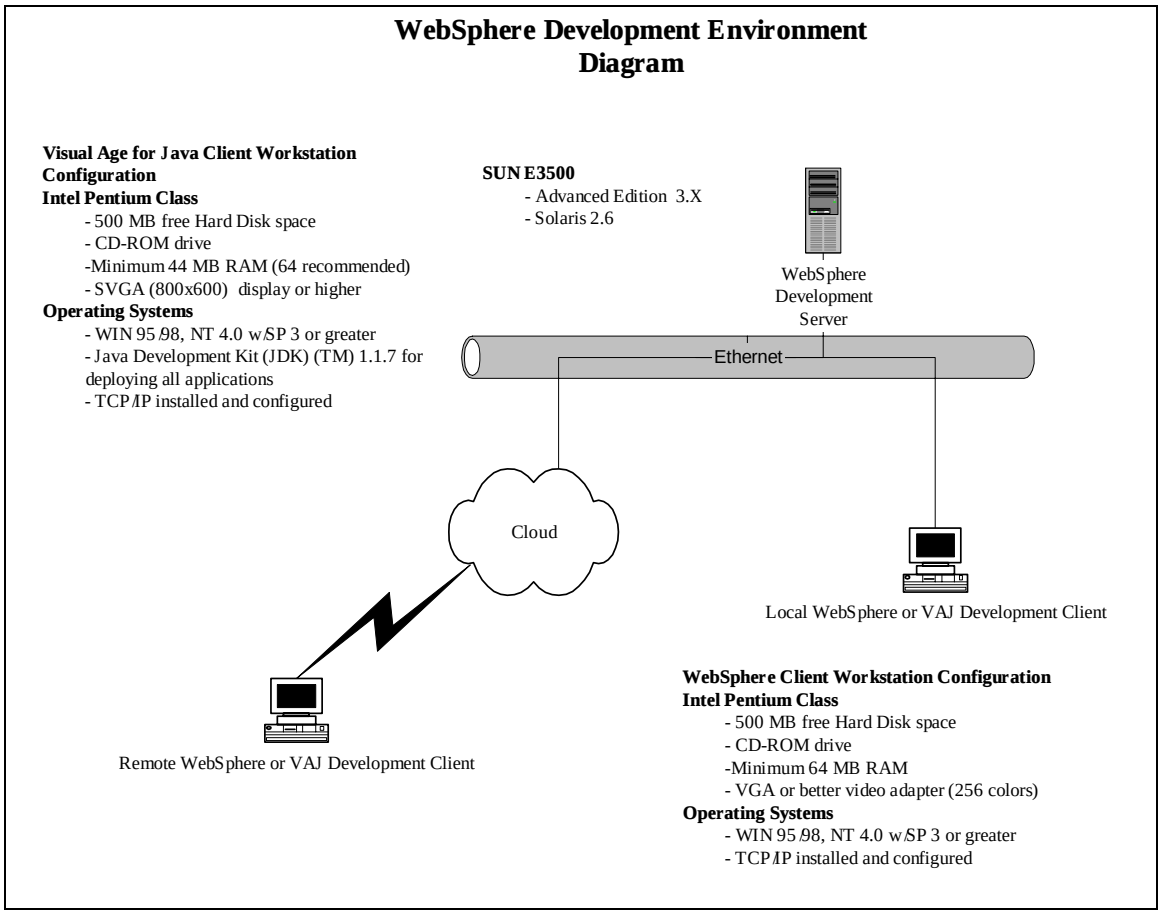

Figure 13 - WebSphere Development Environment

The figure above depicts the development architecture for WebSphere and VisualAge for Java development.

3.4.4. WebSphere Studio Hardware/Software Configurations

Table 12 – WebSphere Studio Version 3.x hardware and software configurations.

Hardware Requirements	Software Requirements
- Intel® Pentium® class PC - 500 MB of free disk space - CD-ROM drive - 64 MB of memory - VGA or better video adapter, configured for at least 256 colors	One of the following operating systems, - Microsoft Windows® 95 - Microsoft Windows 98 - Microsoft Windows NT® Workstation, Server - Version 4.0 with Service Pack 3 - Microsoft Windows 2000 Professional, Server or Advanced Server
	- Microsoft Internet Explorer, V4.0, or higher - Microsoft Internet Explorer, V5.0 is required to use Page Detailer, (a component of Studio)
	TCP / IP installed and configured

Table 13 – Visual Age for Java Hardware and Software Configurations

Hardware Requirements	Software Requirements
- Intel® Pentium® class PC - 500 MB of free disk space - CD-ROM drive - 48 MB of memory (64 recommended) - SVGA	One of the following operating systems, - Microsoft Windows® 95 - Microsoft Windows 98 - Microsoft Windows NT® Workstation, Server - Version 4.0 with Service Pack 3 - Microsoft Windows 2000 Professional, Server or Advanced Server
	- Frames-capable Web browser such as Netscape Navigator 4.04 or higher, or Microsoft (R) Internet Explorer 4.01 or higher
	- Java Development Kit (JDK) (TM) 1.1.7 for deploying all applications
	- TCP / IP installed and configured

Table 14 – WebSphere Advanced Edition 3.0.2 hardware and software configurations.

Hardware Requirements	Software Requirements
- SUN E3500 server - 75 MB of free hard disk space, plus 20 MB for installing IBM HTTP server - 150 MB for installing UDB, metadata repository - CD-ROM drive - 256 MB of memory (512 MB recommended)	- Solaris 2.6 at the latest maintenance level
	- Web browser that supports HTML 4 and Cascading Style Sheets (such as Netscape Navigator 4.07)
	Web Server, IBM HTTP Server Version 1.3.6.2 (available in the WebSphere Application Server package) Apache Server Version 1.3.6 Domino Version 5.0 Lotus Domino Go Webserver Version 4.6.2.5 or 4.6.2.6 Netscape Enterprise Server Version 3.51 or Version 3.6
	- Java Development Kit (JDK) (TM) 1.1.7_08
	- Oracle Version 8.0.5 with Driver Manager JDBC-Thin / 100% Java for JDK1.1.x

3.4.5. WebSphere/VAJ Development tools

Component Broker Tools

The CB Tools can be used with VisualAge for Java and VisualAge C++ to generate multi-tier applications required by the Component Broker Server run time and systems management environments.

Object Builder

Object Builder is the development environment for Component Broker. You can use Object Builder to:

- Develop new applications
- Accumulate existing applications
- Add new functionality to existing applications
- Package an application
- Prepare enterprise beans for execution in the Component Broker run time

The Object Builder is used to develop applications from start to finish. Developers can also start by designing in Rational Rose and then importing the design into Object Builder, and then add the final objects and program logic. Object Builder supports the CORBA programming model using IDL with implementations in both Java and C++. Complete working applications can be generated, including unit test versions and full client/server packages complete with server setup scripts.

The CB Tools integrate with the Object Builder to create the input to compile code and emitters. This integration defines Object Builder as the development environment for the Component Broker product. Applications can be developed from start to finish, or architectural designs can be imported into the Object Builder, where the final objects and program logic can be added.

Developers can set constraints to ensure that the developed components are deployable on the target platforms. These constraints can be set when an object is created, or later by editing its properties. Object-specific platform constraints can be set on:

- Data object implementations
- Managed objects
- Managed object configurations
- Containers
- Dynamic link library (DLL) files

The Object Builder user interface provides access to different views of an application. In order to build the application DLL files defined in the Object Builder, the Component Broker Server Software Development Kit (SDK) must be installed, as well as any prerequisite application development software. The model for the application is constructed from components. Many of the development tasks in Object Builder revolve around defining components.

Component Broker also provides tools for deploying portable enterprise beans to run as Component Broker business objects. Deployment of both session beans and entity beans is supported. Command-line and GUI interfaces for bean deployment are provided, including support for deploying enterprise beans from Object Builder or from VisualAge for Java. Deployment of session beans and entity beans with bean-managed persistence can be done unattended in batch mode or from a makefile. Deployment of entity beans with Connection Management Protocol (CMP) involves using Object Builder to define the mapping between the bean's CMP fields and a persistent data store. This mapping can be done either by using a legacy data store or by defining a new database, and can utilize the full variety of persistent backends that are supported by Component Broker. The mapping can also take advantage of the rich set of database mapping helpers that Component Broker provides.

4 Data Warehouse Architecture Development Environment

Development architecture provides an environment for component-based solutions that supports a team through the Analysis, Design, and Construction phases of the development process. It should also serve as a productive environment for the on-going maintenance of an application.

4.1. Data Warehouse Tools

The tools used as part of the technical architecture to support the data warehouse include the following.

Table 15 – Data Warehouse Tools

Type	Tool	Version	Component
OLAP development	MicroStrategy Agent	7.0	Build
OLAP development	MicroStrategy Architect	7.0	Build
OLAP development	MicroStrategy Administrator	7.0	Build
Web page editor			Build
XML / XSL editor			Build
Design / Modeling			Design
ETL Development	PowerCenter Designer Module PowerCenter Server Manager / Repository Manager	1.7	Build

4.2. Data Warehouse Development Architecture Diagram

4.2.1. Development Environment (MicroStrategy Version 7.0)

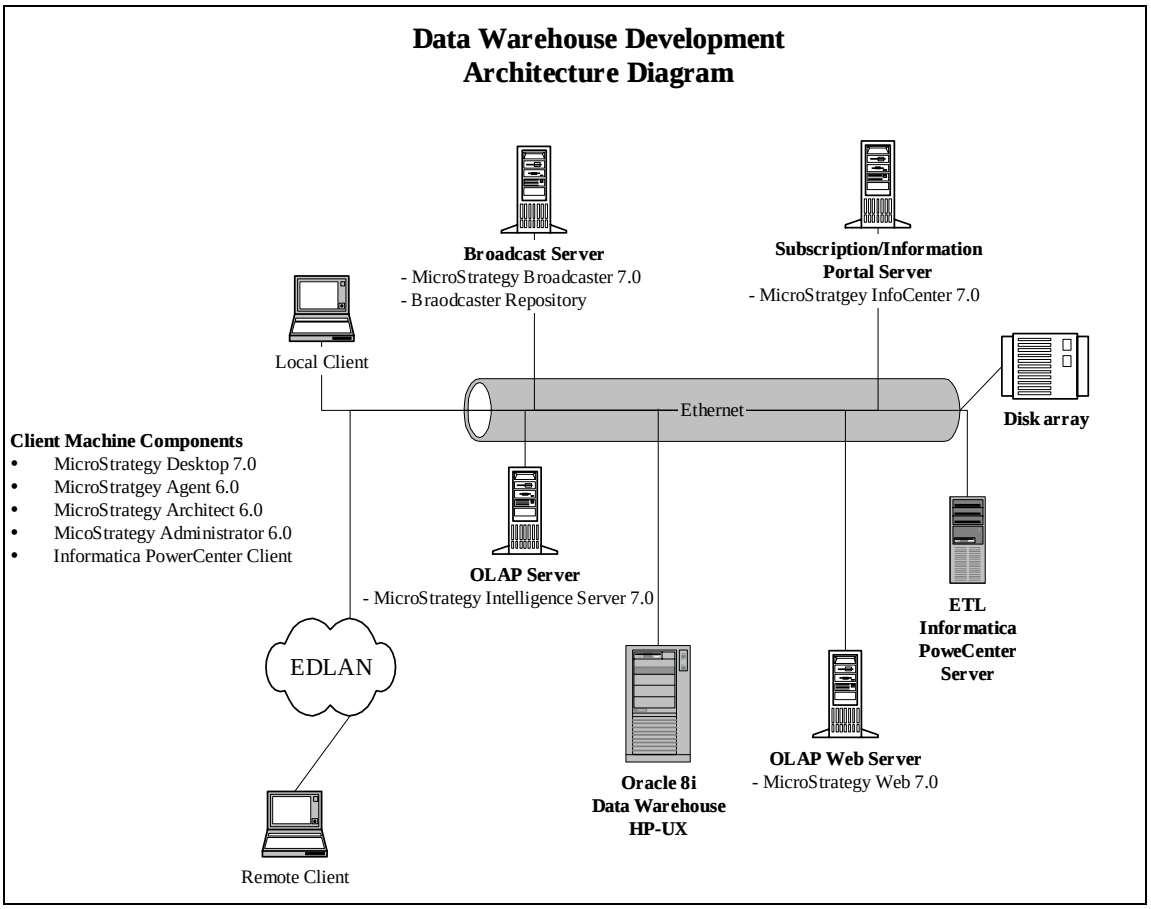

Figure 14 – Data Warehouse Development Architecture

Table 16 – Data Warehouse Development Requirements (7.0)

	Hardware	Machine Type	Machine Function
1.	▪ HP	HP UNIX	Data Warehouse
2.	▪ Dual Processor Pentium PIII, 500 MHz ▪ 1 GB RAM ▪ 15 GB disk space	Windows NT	MicroStrategy Intelligence Server 7.0
3.	▪ Dual Processor Pentium PIII, 500 MHz ▪ 1 GB RAM ▪ 15 GB disk space	Windows NT	MicroStrategy Web 7.0
4.	▪ Dual Processor Pentium PIII, 500 MHz ▪ 1 GB RAM ▪ 15 GB disk space	Windows NT	MicroStrategy InfoCenter 7.0

	Hardware	Machine Type	Machine Function
5.	▪ Dual Processor Pentium PIII, 500 MHz ▪ 1 GB RAM ▪ 15 GB disk space	Windows NT	MicroStrategy Broadcaster 7.0, Broadcaster Repository
6.	▪ 2 Gig of RAM or greater ▪ 10 Gig disk space	Sun Solaris 2.6	Informatica
7.	▪ Pentium, 266 MHz ▪ 64 MB RAM (Minimum requirement) ▪ 2 GB disk space (40 MB for installation)	Windows 95/98Windows NT 4.0	MicroStrategy Agent 7.0, MicroStrategy Architect 7.0, MicroStrategy Administrator 7.0 Informatica PowerCenter Client Tools

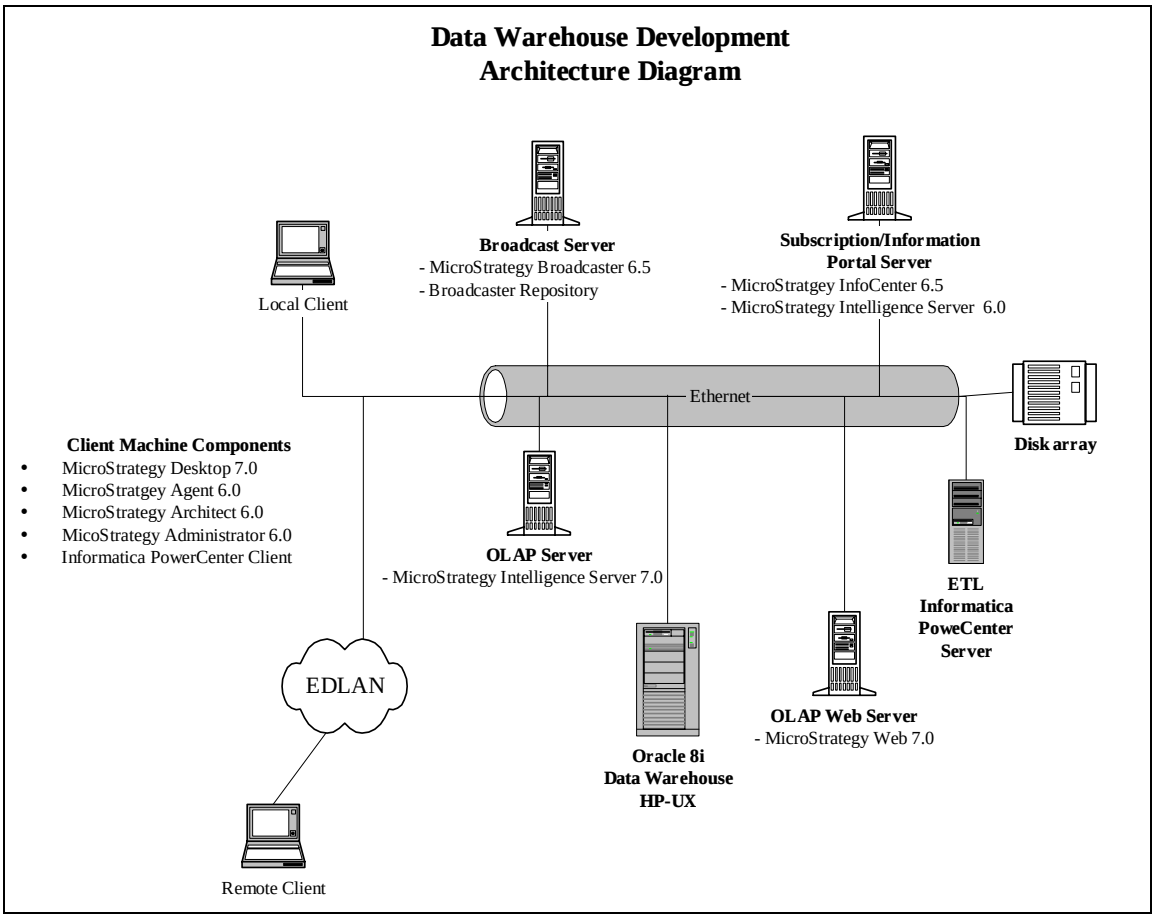

Figure 15 - Temporary Development Environment (MicroStrategy Version 6.5)

Table 17 – Data Warehouse Development Requirements (6.5)

	Hardware	Machine Type	Machine Function
1.	HP UNIX	HP UNIX	Data Warehouse
2.	Dual Processor Pentium PIII, 500 MHz 1 GB RAM 15 GB disk space	Windows NT	MicroStrategy Intelligence Server 7.0
3.	Dual Processor Pentium PIII, 500 MHz 1 GB RAM 15 GB disk space	Windows NT	MicroStrategy Web 7.0
4.	Dual Processor Pentium PIII, 500 MHz 1 GB RAM 15 GB disk space	Windows NT	MicroStrategy InfoCenter 6.5, MicroStrategy Intelligence Server 6.0
5.	Dual Processor Pentium PIII, 500 MHz 1 GB RAM 15 GB disk space	Windows NT	MicroStrategy Broadcaster 6.5, Broadcaster Repository
6.	2 Gig of RAM or greater 10 Gig disk space	Sun Solaris 2.6	Informatica
7.	Pentium, 266 MHz 64 MB RAM (Minimum requirement) 2 GB disk space (40 MB for installation)	Windows 95 / 98Windows NT 4.0	MicroStrategy Agent 7.0, MicroStrategy Architect 7.0, MicroStrategy Administrator 7.0; MicroStrategy Agent 6.0, MicroStrategy Architect 6.0, MicroStrategy Administrator 6.0, Informatica PowerCenter Client Tools

4.3. Informatica Development

4.3.1. Informatica PowerCenter Designer Module

The Designer tool is used for developing Extract, Transform, Load (ETL) capabilities. The tool typically is loaded on a developers workstation (i.e. client workstations). The tool allows the developers to build the mappings, which are the cornerstone of ETL process.

The Designer is comprised of four integrated modules: Source Analyzer, Warehouse Designer, Mapping Designer and Transformation Developer. This is the key tool used by Data Integrator Developers. Mappings typically are setup to export the data from source tables, in this case stored on mainframe DB2 tables, modify the data using business logic, and

then loads the data into target tables, in this case Oracle tables, for later retrieval, querying, and reporting.

4.3.2. Informatica PowerCenter Server Manager Module

The Server Manager is a client tool used to schedule, execute, and monitor sessions and batches that perform the source to target data loads. The Server Manager allows users to navigate through multiple folders and repositories, and monitor multiple Informatica Servers. The tool is a scheduler window for the different sessions and batches, which are typically composed of mappings created in designer. The Server Manager gives the data warehouse administrator control of the extracting and loading process. With the Server Manager the administrator can setup the production and / or test load window of the data warehouse. In many cases this window is overnight when server and database usage is light. The Server Manager will be used in the ITA to setup the ETL window, schedule ad hoc jobs, and manage the development and test loads. Some of the features of the Server Manager:

4.3.3. Informatica PowerCenter Repository Manager Module

The PowerCenter Repository is the metadata integration hub of PowerCenter 1.7. Users gain access to the metadata stored in the repository through the Repository Manager and the Metadata Browser. The Repository Manager is used to create and maintain the PowerCenter repository and its metadata. Within the repository are three levels of detail:

- **Folders**—The highest-level logical groupings of data. Examples are Customer, Vendor, Product, and Human Resources, permit sharing through Shared Folders, which may contain custom groupings of shareable transformations.
- **Mappings**—The business rules that source data must follow as it populates the data mart . One or more mappings may be included within a Folder. Examples are fact table loads, dimension table loads, and aggregate table calculations.
- **Objects** -- The lowest level of detail with which users interact. Examples are source tables, data mart tables and transformations.

4.4. Data Warehouse Development

In order to develop the data warehouse schema, tables, indexes, and other database objects, it is recommended that a data-modeling tool like ERwin be used. These tools will be used to create the logical model, physical model, and database scripts needed to create the data warehouse.

4.5. MicroStrategy Development

The following tools are required to develop application with MicroStrategy products:

4.5.1. MicroStrategy Architect

MicroStrategy Architect is used to create a MicroStrategy project. It is used to connect to the metadata repository and data warehouse and to identify all of the data warehouse tables,

attributes, facts, and relationships. Developers will assign business terms to each element in the data warehouse and identify attribute hierarchies. To create the necessary level of security, the developer must also create user roles, assign users to groups, and assign privileges to schema objects.

4.5.2. MicroStrategy Agent

Once MicroStrategy Architect creates a project, MicroStrategy Agent is used to create all of the reporting objects. Reporting objects include temples, filters, metrics, reports, and documents. Developers should create all reporting objects necessary to satisfy user requirements. All reporting object that are created should be located in the appropriate user folder, typically Public, so that all users can access the objects.

4.5.3. MicroStrategy Administrator

MicroStrategy Administrator is used to manage the reporting objects in the development, test, and production environments and between users. Once reporting objects have been developed, an administrator or developer can move the objects to an identical project in a test environment. Once testing is complete, reporting object can then be moved into production. If a user has created a report that may be useful to all users, an administrator or developer can move or copy the reporting objects to the Public folder thus giving all users access.

4.5.4. Web Page Editor

Web development may be necessary for MicroStrategy Web, MicroStrategy InfoCenter, and MicroStrategy Broadcaster. MicroStrategy Web can use used with the out-of-the-box interface. However, most applications will require some level of customization. Organizations typically request to have web applications utilize the same color scheme and page layout. In order to customize the MicroStrategy Web interface, web development is necessary.

MicroStrategy InfoCenter also has an out-of-the-box interface. Typically, organizations will want to customize this interface so that it is consistent with other web applications. Web development would be required for MicroStrategy InfoCenter customization.

MicroStrategy Broadcaster can send out reports and information to users via HTML e-mail. MicroStrategy Broadcaster does come with a basic web editor to create these e-mails but more sophisticated e-mail layouts may require a more robust web development tool.

4.5.5. XML / XSL Editor

MicroStrategy Agent, MicroStrategy Web, and MicroStrategy Broadcaster use Extensible Markup Language (XML) / Extensible Stylesheet Language (XSL) to format reports. In order to create a custom report layout, the creation or modification to XSL style sheets is required. An XML / XSL editor is recommended for XSL development.

5 EAI Development Architecture – MQSeries Products

The development environment for the EAI is performed in a Microsoft Windows environment. The products and components are loaded on one or two Windows NT servers. Products will be installed as a two-tier environment as there is no dedicated database server. The database utilized for the EAI development architecture consists of an internal DB2 database that is included with the installation media and is configured during the initial installation. This database is used for storing internal MQ product configuration metadata.

The products deployed in the Integrated Technical Architecture supporting the EAI are the IBM MQSeries family products, consisting of the following offerings:

- IBM MQSeries Messaging
- IBM MQSeries Integrator Version 2.0
- IBM MQSeries Workflow

5.1. IBM MQSeries Messaging Introduction

MQSeries provides assured, once-only delivery of messages between IT systems. MQSeries provides support for applications, described below,

- Application programming interfaces: the *Message Queue Interface (MQI)* and *Application Messaging Interface (AMI)* are supported in several programming languages.
- Communication models: *point-to-point* (including *request/reply* and *client/serve*r) and *publish/subscribe* are supported.
- The complexities of communications programming are handled by the messaging services and are therefore removed from the application logic.
- Applications can access other systems and interfaces through gateways and adapters to products such as Lotus Domino, Microsoft Exchange / Outlook, SAP / R3, and IBM's CICS and Information Management System (IMS) / External System Architecture (ESA) products.

5.2. IBM MQSeries Integrator V2.0 Introduction

MQSeries Integrator works with MQSeries messaging, extending its basic connectivity and transport capabilities to provide a powerful *message broker* solution driven by business *rule*s. Messages are formed, routed, and transformed according to the rules defined by an easy-to-use graphical user interface.

Diverse applications can exchange information in unlike forms, with brokers handling the processing required for the information to arrive in the right place in the correct format, according to the defined rules. The applications have no need to know anything other than their own conventions and requirements.

Applications also have much greater flexibility in selecting which messages they wish to receive, because they can specify a *topic* filter, or a *content-based filte*r, or both, to control the messages made available to them. MQSeries Integrator provides a framework that supports supplied, basic, functions along with *plug-in* enhancements, to enable rapid construction and modification of business processing rules that are applied to messages in the system.

5.3. IBM MQSeries WorkFlow Introduction

MQSeries Workflow works with MQSeries messaging by aligning and integrating an organization's staff resources, processes, and capabilities with business strategies. It enables the organization to accelerate process flow, optimize costs, eliminate errors and improve workgroup productivity. MQSeries Workflow is designed to enable integration of all participants in the business process. It ensures the right information gets to the right person at the right time. MQSeries Workflow can be used in combination with MQSeries Integrator, providing a high level of flexibility to allow business and message processing to be as simple or as complicated as the business demands.

5.4. EAI Development Architecture Diagram

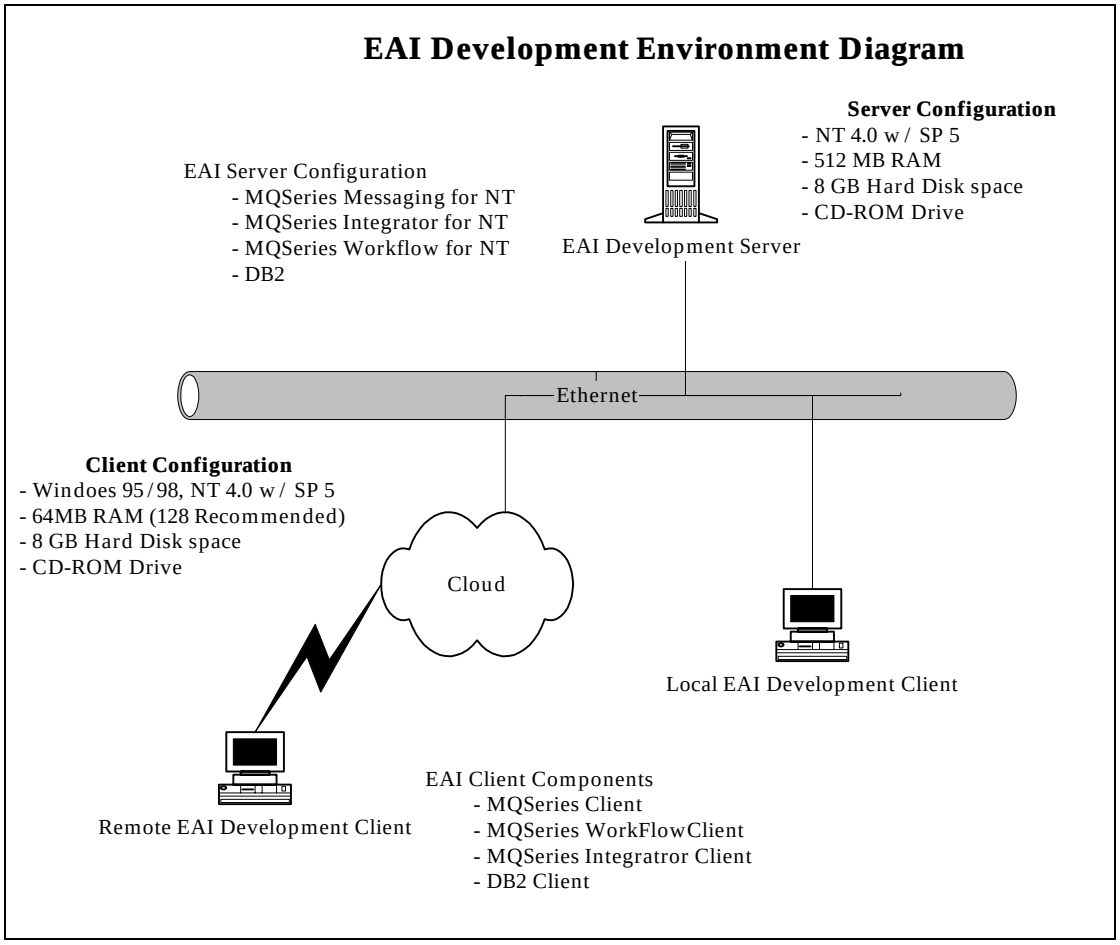

Figure 16 – EAI Development Environment

5.5. IBM MQSeries Messaging Development Environment

MQSeries is a communications system that provides assured, asynchronous, once-only delivery of data across a broad range platforms. Components of the MQSeries product include a Server and Client installation. An MQSeries server is a full queue manager, which can accept MQI calls directly from application programs that are running on the server processor; in addition, it can accept MQI requests from MQSeries clients.

An MQSeries client is part of the MQSeries Messaging product that can be installed on a machine without installing the full queue manager. It accepts Message Queue Interface (MQI) calls from application programs and passes MQI requests to a server that is usually executing on another processor.

The three fundamental concepts in MQSeries are:

- Messages
- Queues
- Queue managers

5.5.1. Messages

A message is a string of bytes that has meaning to the applications that use it. Messages are used for transferring data from one application to another (or to different parts of the same application). The applications can be running on the same platform, or on different platforms.

MQSeries messages have two parts; the application data and a message descriptor. The content and structure of the application data is defined by the application programs that use the data. The message descriptor identifies the message and contains other control information, such as the type of message and the priority assigned to the message by the sending application.

5.5.2. Queues

A queue is a data structure in which messages are stored. The messages may be put on, or got from, the queue by applications or by a queue manager as part of its normal operation. Queues exist independently of the applications that use them. A queue can exist in main storage (if it is temporary), on disk or similar auxiliary storage (if it must be kept in case of recovery), or in both places (if it is currently being used, and must also be kept for recovery). Each queue belongs to a queue manager, which is responsible for maintaining it. The queue manager puts the messages it receives onto the appropriate queue. Queues can exist either in your local system, in which case they are called local queues, or at another queue manager, in which case they are called remote queues.

Applications send to, and receive messages from, queues. For example, one application can put a message on a queue, and another application can get the message from the same queue.

Each queue has queue attributes that determine what happens when applications reference the queue. The attributes indicate:

- Whether applications can retrieve messages from the queue (get enabled)
- Whether applications can put messages onto the queue (put enabled)
- Whether access to the queue is exclusive to one application or shared between applications
- The maximum number of messages that can be stored on the queue at the same time (maximum queue depth)
- The maximum size of messages that can be put on the queue (maximum message size)

5.5.3. Queue managers

A queue manager provides queuing services to applications, and manages the queues that belong to it. It ensures that:

- Object attributes are changed according to the details received.
- Special events (such as instrumentation events or triggering) are generated when the appropriate conditions are met.
- Messages are put on the correct queue, as requested by the application. The application is informed if this cannot be done, and an appropriate reason code is given.

Each queue belongs to a single queue manager and is said to be a local queue to that queue manager. The queue manager, to which an application is connected, is said to be the local queue manager for that application. For the application, the queues that belong to its local queue manager are local queues. A remote queue is a queue that belongs to another queue manager. A remote queue manager is any queue manager other than the local queue manager. A remote queue manager may exist on a remote machine across the network or it may exist on the same machine as the local queue manager. MQSeries supports multiple queue managers on the same machine.

5.6. IBM MQSeries Integrator Development Environment

MQSeries Integrator Version 2.0 is a *message broker* product, addressing the needs of business and application integration through management of information flow. It can route a message to several destinations based on message content rules and transform messages from one format to another depending on the sending and receiving applications requirements. Messages, or part of messages, can be stored in a database, and message contents can be modified before transferring to a final destination. Messages may also be published, making them available to other applications. These services are based on the messaging transport layer provided by the MQSeries products. Communications between all components in the broker domain are provided by MQSeries Messaging.

MQSeries Integrator has four major components that may be installed separately and independently, or all together, depending on the available environment.

5.6.1. Broker

The broker is a named resource that hosts and controls the business processes, which is defined as *message flow*s. Applications communicate with the broker to take advantage of the services provided by the message flows. Applications send new messages to the message flow, and receive processed messages from the message flow, using MQSeries queues and connections. Any number of brokers can be installed, created, and started within a broker domain. More than one broker can be created on a physical system, but a unique queue manager for each broker must be specified. However, a single broker can share a queue manager with the *Configuration Manager*. When creating a broker, the following resources are also created:

- A set of tables in a database to hold the broker's local data. The broker uses an ODBC connection to the database. These broker tables are also referred to as the broker's local persistent store.
- A set of fixed-name queues, defined to the queue manager that hosts this broker. This queue manager must be identified when the broker is created, and it must exist on the same physical system as the broker. It is created when the broker is created, if it does not already exist.

When creating a broker on the system on which the broker component has been installed, the information about the broker's configuration is not automatically recorded in the configuration repository (managed by the Configuration Manager). The Control Center (the *Topology* view) is used to create a reference to this broker with the same name that was specified when the broker was created.

Creating a reference:

- Stores the broker information in the configuration repository.
- Defines a default *execution group* on this broker. Further execution groups can be defined if chosen. Each message flow providing a service on this broker must be deployed to an execution group before that service can be used by applications.

When creating the broker reference, the changes to the broker domain must be *deployed* for them to take effect.

The deploy action:

- Initiates communications between the Configuration Manager and the broker.
- Initializes the broker so that it is ready to execute message flows. The broker receives configuration information from the Configuration Manager, and stores it in its database.

When creating the broker reference, message flows can be assigned to the broker's execution groups, and any message sets required by those message flows to the broker. These changes must also be deployed before they can be activated. These resources can be deployed individually, or together, but until all related resources (for example, a broker, a message flow and the message set it uses) are deployed, the message flow cannot be used on that broker.

When designing the broker domain, the services need to be defined and supported by the brokers. Each service is rules-based and performs a set of actions on each message received. Each action, or actions, is grouped together in a sequence to form a *message flow*. Message flows are created using the *Control Center*.

5.6.2. Configuration Manager

The *Configuration Manager* is the main component of your MQSeries Integrator environment. The components and resources managed by the Configuration Manager constitute the *broker domai*n. The Configuration Manager serves three main functions:

- It maintains configuration details in the *configuration repositor*y. This is a set of database tables that provide a central record of the broker domain components.
- It manages the initialization and deployment of brokers and message processing operations in response to actions initiated through the *Control Cente*r. It communicates with other components in the broker domain using MQSeries transport services.
- It checks the authority of defined user IDs to initiate those actions.

To manage your broker domain a single Configuration Manager must be installed, created, and started. Once started, the Configuration Manager runs in the background.

Using the Control Center the contents of the configuration repository can be viewed, created, modified, and deleted.

The Configuration Manager provides a service to the other components in the broker domain, providing them with configuration updates in response to actions taken by the Control Center. The Configuration Manager validates that the user requesting each action from the Control Center is authorized to perform that action.

The Configuration Manager requires an MQSeries queue manager, and access to a database to maintain internal data in tables. One set of tables holds configuration and definition information for the whole broker domain, and is known as the configuration repository. The second set of tables holds definition information for messages defined or imported through the Control Center, and is known as the message repository.

5.6.3. Control Center

The Control Center interacts with the Configuration Manager to allow the configuration and controlling of the broker domain. The Control Center and Configuration Manager exchange messages (using MQSeries) to provide the information requested, and to make updates to the broker domain configuration. Any number of Control Center instances can be installed and invoked. The Control Center is structured as a number of views on the configuration and message repositories. Users can choose which set of the views are currently included by selecting one of five roles, one of which, “All roles”, shows every view. Within the boundaries of what each user is authorized to do, the Control Center allows the user to retrieve information selectively from:

- The message repository. This contains all message definitions the user (or any other user) has created or imported through the Control Center.

- The configuration repository. This contains configuration information pertaining to all other resources within the broker domain: brokers, collectives, message processing nodes, message flows, topics, and subscriptions.

The Control Center provides the following,

- Develop, modify, assign, and deploy message flows.
- Develop, modify, assign, and deploy message sets.
- Define the broker domain topology and create collectives.
- Control topic security of messages by topic.
- View status information.

To make any changes, the user must *check out* (request a locked copy of) the resource requiring update. This allows updates to the central data to be coordinated by the Configuration Manager. The Control Center shows the users which resources are currently checked out. Once a resource is locked, the user has exclusive control over it until it is returned to the configuration repository using *Check in*, or until the user relinquishes control by unlocking it. Once changes have been made, or a new resource is created, the user can save a local copy. The resource can also be checked in to save the changes in the message or configuration repository, if the user is authorized to do so. This makes changes visible to all other users of the Control Center. Once a decision is made as to which message flows and message sets should reside in each broker, they can be assigned from the *Assignment* view. Following the assignment of these resources, these changes must be deployed through the broker domain. Deployment results in the Configuration Manager sending messages and information about the changes made to the brokers. The *Operations* view and the *Log* view can be used to monitor the success and progress of this step.

5.6.4. User Name Server

The User Name Server is created to handle information requests from the *brokers*. A User Name Server is created as an NT service that is seen to be running using the NT Task Manager. One User Name Server is to be created and configured within each broker domain. The User Name Server does not require database access, but it does rely on a dedicated MQSeries queue manager.

5.7. IBM MQSeries Workflow Development Environment

MQ Workflow provides a way to model a process and assign applications to activities in the resulting workflow model. This enables the workflow manager to automate the control of activities and the flow of data. Work can be routed to the person who performs the activity instance. An application program required to perform an activity instance can be designed to start when a user starts an activity instance.

An MQ Workflow system has a tiered structure:

- **Tier 1 — Client tier**

Tier 1 contains the MQ Workflow system clients, application programming interfaces, and Buildtime. They use the MQ client manager and client message layer to connect with the second tier. The administration utility sits within this tier and uses the administration API (application programming interface) to communicate with the administration server located in tier 2.

- **Tier 2 — Server tier**

  Tier 2 contains the group of servers that are collectively referred to as the MQ Workflow server. This is the working center where all the scheduling, distribution, cleanup, administration, server communication, and execution is done. The administration server is located in this tier. It communicates with the administration utility located in tier 1 and all other components in the system using MQSeries techniques.

- **Tier 3 — Database tier**

  Tier 3 contains the MQ Workflow relational database. This holds system status and setup information and stored procedures for the complete system. The administration server accesses system tables within the database and returns the contents back to the administration utility when requested to do so. The database is automatically updated by the administration server in response to system events.

The Workflow product can be divided into three functional component groups:

- Server Components
- Buildtime Components
- Client Components

5.7.1. Server Components

The server components coordinate and manage an MQ Workflow system and its clients. Server components are also responsible for keeping track and administering process execution.

Execution Server

The Execution Server is responsible for moving the right work item to the right person at the right time. To achieve this, the Execution Server performs the following tasks:

- Interpreting the process definitions, that is, definitions for staff, programs, and data
- Creating the process instances and managing their execution, including starting, stopping, or suspending them
- Navigating between activities and creating the work items needed for processing
- Managing process states and logging events
- Maintaining the worklists of Runtime users

The Execution Server acts as a Database Client and communicates with the Database Server.

Administration Server

The Administration Server manages an MQ Workflow system. The Administration Server communicates with all other components in a system or system group. It is the working center of the administration component. The Administration Server is responsible for the availability, operation, and error recovery of all server components. The Administration Server uses its self-recoverable feature to guarantee system consistency and operation. If a user needs to access the Administration Server, MQ Workflow offers an Administration Utility.

Scheduling Server

The Scheduling Server controls and manages notification for activities that must be performed within a certain time frame. If items are overdue for a process, the Scheduling Server sends notifications to the worklists of the relevant persons.

Cleanup Server

The Cleanup Server is responsible for physically deleting process instances that are finished. Depending on the definitions you set for the system in Buildtime, finished processes are deleted immediately or later in the day when the system is idle.

Java CORBA Agent

The Java CORBA Agent is responsible for routing CORBA IIOP requests from the Java API to the Execution Server and sending responses back.

5.7.2. Buildtime Components

With Buildtime the user can create workflow models and define system resources. Buildtime offers a graphical editor for creating process models. Other features in Buildtime allow the user to define your organization and the programs to be used in the workflow as well as the network definitions. Existing workflow models can be imported into MQ Workflow or exported to the MQ Workflow Definition Language (FDL). Workflow models can also be exported in HTML. Once a decision is made as to which workflow model to use, the user translates the model into a template that can be started from an MQ Workflow Client and managed by the server components.

Modeling processes with MQ Workflow Buildtime

The first stage of using MQ Workflow is to build a workflow model representing the 'real' processes of your enterprise. Process models define, for example:

- Work items in the process and the order in which they occur
- Staff assigned to manage and perform each work item
- Process-relevant data used in each work item and passed to subsequent ones
- Programs needed to perform work items

- Conditions for starting and ending each work item
- Maximum duration of each work item and process

The Role of the Programmer in Modeling a Process

As workflow models are defined, the applications and data structures needed to support program activities are identified. Programmers can create new applications, integrate existing applications, or reengineer existing applications to support these program activities. To reengineer existing applications with the workflow model, programmers must determine if the applications used by the enterprise can be functionally decomposed. The control and flow logic are separated from the application, the start and exit conditions are moved into the workflow model, and the program is divided into modules to be invoked by the workflow manager at the appropriate points. The resulting modules are applications that are assigned to perform the program activities defined in the workflow model.

Guidelines for drawing a Process Diagram

When drawing the process diagram, be sure to sequence the activities in the diagram carefully.

1. Draw control connectors to show the order in which activities are to be performed.
2. Draw data connectors to show where output data from an activity is required as input to a later activity. Or draw a data connector as a loop originating from and directed back to the same activity or block if you want output data mapped to input data for repeated executions of the activity or block. Input data can also be mapped to output data.
3. Make sure that each data connector between two activities has a corresponding control connector.
4. Do not draw a control or data connector from a later activity to a previous activity. MQ Workflow prevents the accidental creation of such cycles in the diagram.
5. If there are a series of activities that should be repeated, put these within a block and specify an exit condition.
6. If the diagram is large or complex, consider using subprocesses to simplify its appearance and reflect orderly levels of complexity. To reuse a set of activities in other processes, define these activities in a subprocess. Use blocks if there is a set of activities that must be repeated until an exit condition is met. The block acts as a do-until loop.
7. When drawing a process activity that starts a process, which contains other process activities, check the sequences of these calls carefully. A process can start instances of other processes in any order, and can start other instances of itself.

To help make the diagram neat, position the activities using a grid on the drawing area. In the **Format** menu, click **Grid** and select **Snap to Grid**.

Prerequisites for programming language API

MQ Workflow application development assumes that the appropriate environment is established. This means that:

- The MQ Workflow Development Kit is installed on the machine where application development is occurring.
- A compiler of one of the supported languages is installed and configured. (C, C++, Active X Controls and Object Linking & Embedding (OLE) Objects, Java, XML messages, Lotus Notes)

The basic interfaces for requesting Runtime services from MQ Workflow are a C-language API and an XML message interface.

5.7.3. The Client Components

The MQ Workflow Client starts processes and monitors their execution. The Administration Utility administers the system and the Program Execution Agent invokes application programs that are used in the workflow.

Running Processes with the MQ Workflow Client

A modeler defines the workflow model with Buildtime using a graphical user interface. When the model is completed, it must be exported from Buildtime and imported into Runtime. A process model must then be imported and translated into a Runtime *process templat*e. Before working with the processes using the Client, create a runnable copy of a process template. Such a copy is called *process instanc*e, which can then be started.

Working with the Client

When working with the MQ Workflow Client, the following tasks on worklists can be performed,

- Start activities on a worklist
- Select how to filter and sort items in worklists
- Create and delete worklists
- Force a change to the status of an activity
- Work on notified activities and processes
- Delete finished activities
- Monitor the progress of activities within a process instance

In addition to using worklists, you can intervene in the workflow and, for example, change the status of an activity. You can also work with process templates, which represent the workflow model created by a modeler in Buildtime. To start a process, create a process instance from a template.

Starting the Client

To start the Client and to display the **Logon** window, do the following:

1. On the **Start** menu, click **Programs**.
2. Click **MQSeries Workflow**.
3. Then click **MQSeries Workflow Client**.
4. The **Logon** window appears.

The alternatives for you to log on to the Client are as follows:

- Depending on the MQ Workflow server installation, the developer can use unified logon for the MQ Workflow Client in the Windows environment.
- Enter your user ID and your password. You do not have to enter the default system or the system group. For the initial logon, consult the administrator to find out what to enter in the fields.

6 Other Development Tools Product Descriptions

6.1. CCC Harvest

Product Focus - Application lifecycle, software change and configuration management for client / server development environments

PLATINUM CCC / Harvest provides comprehensive change and configuration management solutions for cross-platform, client / server development environments. CCC / Harvest allows users to create a repeatable process through which they manage application development. It is ideal for organizations needing an enterprise-wide software change management solution. CCC / Harvest allows users to create a repeatable process through which they manage application development. CCC / Harvest integrates advanced configuration management with change process automation and problem tracking, providing users with unprecedented visibility and control over the entire software development and maintenance lifecycle. CCC / Harvest also offers automated build support, numerous supplied "out of the box" lifecycle models, extensive reporting capabilities, and supports multi-site development, including distributed server support and automated multi-site synchronization.

6.2. CONTROL-SA

The CONTROL-SA product provides IT organizations with tight control over diverse IT security environments. The product achieves effective comprehensive, enterprise-wide security management from a central point of control. With CONTROL-SA an enterprise can establish and enforce efficient enterprise-wide policies and standards regarding user access to IT resources.

- Centralizes management of users, resource access and security policies through a dedicated security data repository and GUI
- Provides security administrators with a complete view of each user's access privileges
- Introduces role-based management to simplify and automate creating, setting up and revoking user access rights across heterogeneous platforms and applications
- Audits access rights, eliminating security exposures and reducing administrative errors
- Accommodates rapidly changing IT resources and increasing demand for third-party access through an open, expandable architecture

6.3. NetMonitor

NetMonitor is a small application that monitors Internet connections. It shows kilobytes received and sent and graphically represents transfer rates in real time. It also features a keep alive function that prevents the ISP from pulling the plug for inactivity, and allows the conduction of a simple test to determine connection speed.

6.4. Checkpoint Firewall-1

Check Point FireWall-1 is a fully integrated enterprise security suite that includes access control, user authentication, network address translation, content security, auditing, third-party device management, and more - all of which are managed via a single enterprise policy from a central console.

6.5. Rational Suite

Rational Suite Enterprise Version 2000 is designed for project leaders and other practitioners who span the functional boundaries found in software development organizations. Rational Suite Enterprise provides every tool and process in the Rational arsenal for organizations developing and deploying software for e-business, e-infrastructure and e-devices. Projects are expedited by using completely integrated tools that unify the team, optimize individual productivity, and simplify adoption of development best practices.

Rational ClearQuest

Rational ClearQuest is a highly flexible defect and change tracking system that captures and tracks all types of change, for any type of project. The fully customizable interface and workflow engine enable ClearQuest to adapt to any development process, and with support for industry standard databases, ClearQuest scales to support projects of any size. Integration with other development solutions, including configuration management, automated testing and requirements management tools, ensures that all members of your team are tied into the defect and change tracking process.

Rational ClearCase

Rational ClearCase is a configuration management solution. It works seamlessly with Rational ClearQuest to deliver integrated configuration and change management functions that simplify the process of change. Together, these tools offer Unified Change Management, Rational's out-of-the-box workflow for automating change across the e-development life cycle. Rational ClearCase can help streamline the development to complete projects fast and right by providing the following capabilities,

- Accelerate team development across diverse users, platforms and geographically-distributed sites
- Unify change management across the e-development lifecycle
- Manage content and code to eliminate e-development chaos
- Scale from small teams to the enterprise

Rational AnalystStudio

Rational Suite AnalystStudio Version 2000 provides integrated tools and techniques to help analyze business processes, data, and systems before beginning development work.

Rational RequisitePro

RequisitePro is a complete requirements management solution.

Rational Rose

Rational Rose unifies all the e-development teams through modeling - modeling that is based on the Unified Modeling Language (UML). The UML is the standard notation for software architecture. This allows the entire enterprise-wide team to communicate with one language and one tool.

Rational ClearDDTS

Rational ClearDDTS is a change request management product for UNIX specifically designed to track and manage product defects and enhancement requests uncovered during product development and quality assurance testing. It can be tightly integrated with software configuration management products, such as ClearCase, to help users effectively manage change throughout the software development lifecycle.

Rational Suite TestStudio

Rational Suite TestStudio Version 2000 provides automated reliability and functionality testing. It can be used in Java applets and applications. TestStudio makes it easy not only to create and maintain reusable test scripts, but also to test each iteration from early development forward, helping to avoid expensive changes at the end of the development cycle.

Rational Suite PerformanceStudio

Rational Suite PerformanceStudio Version 2000 provides scalable and easy-to-use performance testing tools that can be used in every phase of e-development. The tools supports testing early in the development lifecycle to help reduce project risk and avoid pre-release crises.

Rational TestFactory

Rational TestFactory is an Automated Software Quality tool that does for testers what the word processor did for writers. TestFactory finds reliability defects in applications automatically, without recording or scripting. The quality gap represents the crucial improvement in the quality of an application that occurs after features have been coded but before the product is ready to release. This interval directly impacts the release date and is where defects are found and removed.

Rational SiteCheck

Rational SiteCheck automatically checks your Web site content with every update, and ensures that the latest changes have not adversely affected the site's links. Rational SiteCheck is a tool that will help eliminate unnecessary HTTP errors and reduce user frustration. It is packaged as part of Rational Suite TestStudio and Rational TeamTest.

6.6. Microsoft Office

Microsoft Office Standard edition is designed for users who require only the core desktop productivity tools. It provides tools for creating, publishing, and analyzing information regardless of where it is located. It includes word processing and spreadsheet capabilities, as well as an e-mail and desktop information management tool. Microsoft Office is for users who do not need to manage and track vital business information and don't want to create and manage Web sites. Microsoft Office includes:

- Word: Word processor
- Excel: Spreadsheet
- PowerPoint: Presentation graphics program
- Outlook: Messaging and collaboration software
- Project: Project management software

7 Acronyms

Table 18 – List of Acronyms

Acronym	Description
AIM	Application Interface Messaging
API	Application Programming Interface
ASP	Active Server Page
CGI	Common Gateway Interface
CICS	Customer Information Control System
CMP	Connection Management Protocol
CPU	Central Processing Unit
DDL	Database Definition Language
DHTML	Dynamic Hypertext Markup Language
DRE	Dynamic Reasoning Engine
EAI	Enterprise Architecture Integration
ESA	Extended System Architecture
FDL	Workflow Definition Language
GB	Gigabyte
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines
IDEA	Integrated Development Environment Architecture
IMS	Information Management System
IP	Internet Protocol
ITA	Integrated Technical Architecture
JSP	JavaServer Pages
MQI	Message Queue Interface
OLE	Object Linking & Embedding
PDF	Portable Document Format

Acronym	Description
QA	Quality Assurance
RAM	Random-access Memory
SAN	Storage Area Network
SDK	Software Development Kit
SFA	Student Financial Assistance
TCL	Tool Command Language
TCP	Transmission Control Protocol
UML	Unified Modeling Language
VDC	Virtual Data Center
VPC	Viador Portal Customization
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language